

# LBM API Reference Manual

## 5.3.6

Generated by Doxygen 1.5.2

Thu Mar 6 13:11:08 2014



# Contents

<b>1</b>	<b>Informatica Ultra Messaging (UM) API</b>	<b>1</b>
<b>2</b>	<b>LBM API Module Index</b>	<b>3</b>
2.1	LBM API Modules . . . . .	3
<b>3</b>	<b>LBM API Data Structure Index</b>	<b>5</b>
3.1	LBM API Data Structures . . . . .	5
<b>4</b>	<b>LBM API File Index</b>	<b>11</b>
4.1	LBM API File List . . . . .	11
<b>5</b>	<b>LBM API Page Index</b>	<b>13</b>
5.1	LBM API Related Pages . . . . .	13
<b>6</b>	<b>LBM API Module Documentation</b>	<b>15</b>
6.1	Add a field to a message . . . . .	15
6.2	Add an array field to a message . . . . .	20
6.3	Add an element to an array field by field index . . . . .	24
6.4	Add an element to an array field by field name . . . . .	29
6.5	Add an element to an array field referenced by an iterator . . . . .	34
6.6	Get scalar field values by field index . . . . .	39
6.7	Get scalar field values by field name . . . . .	44
6.8	Get a scalar field via an iterator . . . . .	49
6.9	Get an element from an array field by field index . . . . .	54
6.10	Get an element from an array field by field name . . . . .	59

6.11	Get an element from an array field referenced by an iterator . . . . .	65
6.12	Set a field value in a message by field index . . . . .	70
6.13	Set a field value in a message by field name . . . . .	75
6.14	Set a field value in a message referenced by an iterator . . . . .	80
6.15	Set a field value in a message by field index to an array field . . . . .	85
6.16	Set a field value in a message by field name to an array field . . . . .	89
6.17	Set a field value in a message, referenced by an iterator, to an array field.	93
6.18	Set an array field element value by field index . . . . .	97
6.19	Set an array field element value by field name . . . . .	102
6.20	Set an array field element value for a field referenced by an iterator . .	107
<b>7</b>	<b>LBM API Data Structure Documentation</b>	<b>113</b>
7.1	lbm_apphdr_chain_elem_t_stct Struct Reference . . . . .	113
7.2	lbm_async_operation_func_t Struct Reference . . . . .	115
7.3	lbm_async_operation_info_t Struct Reference . . . . .	116
7.4	lbm_context_event_func_t_stct Struct Reference . . . . .	118
7.5	lbm_context_event_umq_registration_complete_ex_t_stct Struct Reference . . . . .	119
7.6	lbm_context_event_umq_registration_ex_t_stct Struct Reference . .	121
7.7	lbm_context_rcv_immediate_msgs_func_t_stct Struct Reference . .	123
7.8	lbm_context_src_event_func_t_stct Struct Reference . . . . .	124
7.9	lbm_context_stats_t_stct Struct Reference . . . . .	125
7.10	lbm_ctx_umq_queue_topic_list_info_t Struct Reference . . . . .	129
7.11	lbm_delete_cb_info_t_stct Struct Reference . . . . .	130
7.12	lbm_event_queue_cancel_cb_info_t_stct Struct Reference . . . . .	131
7.13	lbm_event_queue_stats_t_stct Struct Reference . . . . .	132
7.14	lbm_flight_size_inflight_t_stct Struct Reference . . . . .	144
7.15	lbm_hf_sequence_number_t_stct Union Reference . . . . .	145
7.16	lbm_iovec_t_stct Struct Reference . . . . .	146
7.17	lbm_ipv4_address_mask_t_stct Struct Reference . . . . .	147
7.18	lbm_mim_unrecloss_func_t_stct Struct Reference . . . . .	148
7.19	lbm_msg_channel_info_t_stct Struct Reference . . . . .	149

7.20 <code>lbm_msg_fragment_info_t_stct</code> Struct Reference . . . . .	150
7.21 <code>lbm_msg_gateway_info_t_stct</code> Struct Reference . . . . .	151
7.22 <code>lbm_msg_properties_iter_t_stct</code> Struct Reference . . . . .	152
7.23 <code>lbm_msg_t_stct</code> Struct Reference . . . . .	153
7.24 <code>lbm_msg_ume_deregistration_ex_t_stct</code> Struct Reference . . . . .	158
7.25 <code>lbm_msg_ume_registration_complete_ex_t_stct</code> Struct Reference . . . . .	160
7.26 <code>lbm_msg_ume_registration_ex_t_stct</code> Struct Reference . . . . .	161
7.27 <code>lbm_msg_ume_registration_t_stct</code> Struct Reference . . . . .	163
7.28 <code>lbm_msg_umq_deregistration_complete_ex_t_stct</code> Struct Reference . . . . .	164
7.29 <code>lbm_msg_umq_index_assigned_ex_t_stct</code> Struct Reference . . . . .	165
7.30 <code>lbm_msg_umq_index_assignment_eligibility_start_complete_ex_t_stct</code> Struct Reference . . . . .	166
7.31 <code>lbm_msg_umq_index_assignment_eligibility_stop_complete_ex_t_stct</code> Struct Reference . . . . .	167
7.32 <code>lbm_msg_umq_index_released_ex_t_stct</code> Struct Reference . . . . .	168
7.33 <code>lbm_msg_umq_registration_complete_ex_t_stct</code> Struct Reference . . . . .	169
7.34 <code>lbm_rcv_src_notification_func_t_stct</code> Struct Reference . . . . .	171
7.35 <code>lbm_rcv_transport_stats_daemon_t_stct</code> Struct Reference . . . . .	172
7.36 <code>lbm_rcv_transport_stats_lbtpc_t_stct</code> Struct Reference . . . . .	173
7.37 <code>lbm_rcv_transport_stats_lbtrdma_t_stct</code> Struct Reference . . . . .	175
7.38 <code>lbm_rcv_transport_stats_lbtrm_t_stct</code> Struct Reference . . . . .	177
7.39 <code>lbm_rcv_transport_stats_lbtru_t_stct</code> Struct Reference . . . . .	184
7.40 <code>lbm_rcv_transport_stats_t_stct</code> Struct Reference . . . . .	190
7.41 <code>lbm_rcv_transport_stats_tcp_t_stct</code> Struct Reference . . . . .	192
7.42 <code>lbm_rcv_umq_queue_msg_list_info_t</code> Struct Reference . . . . .	194
7.43 <code>lbm_rcv_umq_queue_msg_retrieve_info_t</code> Struct Reference . . . . .	195
7.44 <code>lbm_serialized_response_t_stct</code> Struct Reference . . . . .	196
7.45 <code>lbm_src_cost_func_t_stct</code> Struct Reference . . . . .	197
7.46 <code>lbm_src_event_flight_size_notification_t_stct</code> Struct Reference . . . . .	198
7.47 <code>lbm_src_event_sequence_number_info_t_stct</code> Struct Reference . . . . .	199
7.48 <code>lbm_src_event_ume_ack_ex_info_t_stct</code> Struct Reference . . . . .	201
7.49 <code>lbm_src_event_ume_ack_info_t_stct</code> Struct Reference . . . . .	203

7.50 <a href="#">lbtipc_t_stct Struct Reference</a>	204
7.51 <a href="#">lbtrdma_t_stct Struct Reference</a>	206
7.52 <a href="#">lbtrtm_t_stct Struct Reference</a>	207
7.53 <a href="#">lbtrtr_t_stct Struct Reference</a>	209
7.54 <a href="#">lbttr_t_stct Struct Reference</a>	210
7.55 <a href="#">lbtulb_t_stct Struct Reference</a>	212
7.56 <a href="#">lbtulb_receiver_t_stct Struct Reference</a>	213
7.57 <a href="#">lbtulb_message_info_t_stct Struct Reference</a>	215
7.58 <a href="#">lbtulb_stability_ack_info_t_stct Struct Reference</a>	217
7.59 <a href="#">lbtwakeup_t_stct Struct Reference</a>	219
7.60 <a href="#">lbt_notify_func_t_stct Struct Reference</a>	220
7.61 <a href="#">lbt_send_ex_info_t_stct Struct Reference</a>	221
7.62 <a href="#">lbt_transport_stats_daemon_t_stct Struct Reference</a>	223
7.63 <a href="#">lbt_transport_stats_lbtrpc_t_stct Struct Reference</a>	224
7.64 <a href="#">lbt_transport_stats_lbtrdma_t_stct Struct Reference</a>	225
7.65 <a href="#">lbt_transport_stats_lbtrtm_t_stct Struct Reference</a>	226
7.66 <a href="#">lbt_transport_stats_lbtrtr_t_stct Struct Reference</a>	230
7.67 <a href="#">lbt_transport_stats_lbtrulb_t_stct Struct Reference</a>	233
7.68 <a href="#">lbt_transport_stats_tcp_t_stct Struct Reference</a>	235
7.69 <a href="#">lbt_hash_func_ex_t_stct Struct Reference</a>	236
7.70 <a href="#">lbt_timeval_t_stct Struct Reference</a>	237
7.71 <a href="#">lbt_transport_source_info_t_stct Struct Reference</a>	238
7.72 <a href="#">lbt_icast_resolver_entry_t_stct Struct Reference</a>	241
7.73 <a href="#">lbt_ume_ctx_rcv_ctx_notification_func_t_stct Struct Reference</a>	243
7.74 <a href="#">lbt_ume_rcv_recovery_info_ex_func_info_t_stct Struct Reference</a>	244
7.75 <a href="#">lbt_ume_rcv_recovery_info_ex_func_t_stct Struct Reference</a>	246
7.76 <a href="#">lbt_ume_rcv_regid_ex_func_info_t_stct Struct Reference</a>	247
7.77 <a href="#">lbt_ume_rcv_regid_ex_func_t_stct Struct Reference</a>	249
7.78 <a href="#">lbt_ume_rcv_regid_func_t_stct Struct Reference</a>	250
7.79 <a href="#">lbt_ume_src_force_reclaim_func_t_stct Struct Reference</a>	251
7.80 <a href="#">lbt_ume_store_entry_t_stct Struct Reference</a>	252

7.81 <a href="#">lbt_ume_store_group_entry_t_stct Struct Reference</a>	253
7.82 <a href="#">lbt_ume_store_name_entry_t_stct Struct Reference</a>	254
7.83 <a href="#">lbt_umm_info_t_stct Struct Reference</a>	255
7.84 <a href="#">lbt_umq_index_info_t_stct Struct Reference</a>	257
7.85 <a href="#">lbt_umq_msg_total_lifetime_info_t_stct Struct Reference</a>	258
7.86 <a href="#">lbt_umqmsgid_t_stct Struct Reference</a>	259
7.87 <a href="#">lbt_umq_queue_entry_t_stct Struct Reference</a>	260
7.88 <a href="#">lbt_umq_queue_msg_status_t Struct Reference</a>	261
7.89 <a href="#">lbt_umq_queue_topic_status_t Struct Reference</a>	263
7.90 <a href="#">lbt_umq_queue_topic_t_stct Struct Reference</a>	264
7.91 <a href="#">lbt_umq_ulb_application_set_attr_t_stct Struct Reference</a>	265
7.92 <a href="#">lbt_umq_ulb_receiver_type_attr_t_stct Struct Reference</a>	266
7.93 <a href="#">lbt_umq_ulb_receiver_type_entry_t_stct Struct Reference</a>	267
7.94 <a href="#">lbt_wildcard_rcv_compare_func_t_stct Struct Reference</a>	268
7.95 <a href="#">lbt_wildcard_rcv_create_func_t_stct Struct Reference</a>	269
7.96 <a href="#">lbt_wildcard_rcv_delete_func_t_stct Struct Reference</a>	270
7.97 <a href="#">lbmmmon_attr_block_t_stct Struct Reference</a>	271
7.98 <a href="#">lbmmmon_attr_entry_t_stct Struct Reference</a>	272
7.99 <a href="#">lbmmmon_ctx_statistics_func_t_stct Struct Reference</a>	273
7.100 <a href="#">lbmmmon_evq_statistics_func_t_stct Struct Reference</a>	274
7.101 <a href="#">lbmmmon_format_func_t_stct Struct Reference</a>	275
7.102 <a href="#">lbmmmon_packet_hdr_t_stct Struct Reference</a>	277
7.103 <a href="#">lbmmmon_rcv_statistics_func_t_stct Struct Reference</a>	279
7.104 <a href="#">lbmmmon_src_statistics_func_t_stct Struct Reference</a>	280
7.105 <a href="#">lbmmmon_transport_func_t_stct Struct Reference</a>	281
7.106 <a href="#">lbmpdm_decimal_t Struct Reference</a>	283
7.107 <a href="#">lbmpdm_field_info_attr_stct_t Struct Reference</a>	284
7.108 <a href="#">lbmpdm_field_value_stct_t Struct Reference</a>	285
7.109 <a href="#">lbmpdm_timestamp_t Struct Reference</a>	287
7.110 <a href="#">lbmsdm_decimal_t_stct Struct Reference</a>	288
7.111 <a href="#">ume_block_src_t_stct Struct Reference</a>	289

7.112ume_liveness_receiving_context_t_stct Struct Reference . . . . .	290
<b>8 LBM API File Documentation</b>	<b>291</b>
8.1 lbm.h File Reference . . . . .	291
8.2 lbmaux.h File Reference . . . . .	532
8.3 lbmht.h File Reference . . . . .	536
8.4 lbmon.h File Reference . . . . .	542
8.5 lbmpdm.h File Reference . . . . .	574
8.6 lbmsdm.h File Reference . . . . .	614
8.7 umeblocks.h File Reference . . . . .	670
<b>9 LBM API Page Documentation</b>	<b>675</b>
9.1 LBMMON Example source code . . . . .	675
9.2 LBMMON LBM transport module . . . . .	676
9.3 Source code for lbmonrlbm.h . . . . .	677
9.4 Source code for lbmonrlbm.c . . . . .	680
9.5 LBMMON UDP transport module . . . . .	700
9.6 Source code for lbmonrudp.h . . . . .	701
9.7 Source code for lbmonrudp.c . . . . .	704
9.8 LBMMON CSV format module . . . . .	721
9.9 Source code for lbmonfmtcsv.h . . . . .	722
9.10 Source code for lbmonfmtcsv.c . . . . .	727
9.11 LBMMON LBMSNMP transport module . . . . .	766
9.12 Source code for lbmonrlbmsnmp.h . . . . .	767
9.13 Source code for lbmonrlbmsnmp.c . . . . .	770
9.14 Deprecated List . . . . .	792

## **Chapter 1**

# **Informatica Ultra Messaging (UM) API**

[Browse UM API Functions and constants](#)



# Chapter 2

## LBM API Module Index

### 2.1 LBM API Modules

Here is a list of all modules:

Add a field to a message . . . . .	15
Add an array field to a message . . . . .	20
Add an element to an array field by field index . . . . .	24
Add an element to an array field by field name . . . . .	29
Add an element to an array field referenced by an iterator . . . . .	34
Get scalar field values by field index . . . . .	39
Get scalar field values by field name . . . . .	44
Get a scalar field via an iterator . . . . .	49
Get an element from an array field by field index . . . . .	54
Get an element from an array field by field name . . . . .	59
Get an element from an array field referenced by an iterator . . . . .	65
Set a field value in a message by field index . . . . .	70
Set a field value in a message by field name . . . . .	75
Set a field value in a message referenced by an iterator . . . . .	80
Set a field value in a message by field index to an array field . . . . .	85
Set a field value in a message by field name to an array field . . . . .	89
Set a field value in a message, referenced by an iterator, to an array field. . . . .	93
Set an array field element value by field index . . . . .	97
Set an array field element value by field name . . . . .	102
Set an array field element value for a field referenced by an iterator . . . . .	107



# Chapter 3

## LBM API Data Structure Index

### 3.1 LBM API Data Structures

Here are the data structures with brief descriptions:

<code>lbm_apphdr_chain_elem_t_stct</code> (Structure that represents an element in an app header chain) . . . . .	113
<code>lbm_async_operation_func_t</code> (Structure that holds information for asynchronous operation callbacks) . . . . .	115
<code>lbm_async_operation_info_t</code> (Results struct returned via the user-specified asynchronous operation callback from any asynchronous API) . . .	116
<code>lbm_context_event_func_t_stct</code> (Structure that holds the application callback for context-level events) . . . . .	118
<code>lbm_context_event_umq_registration_complete_ex_t_stct</code> (Structure that holds information for contexts after registration is complete to all involved queue instances) . . . . .	119
<code>lbm_context_event_umq_registration_ex_t_stct</code> (Structure that holds queue registration information for the UMQ context in an extended form) .	121
<code>lbm_context_rcv_immediate_msgs_func_t_stct</code> (Structure that holds the application callback for receiving topic-less immediate mode messages) . . . . .	123
<code>lbm_context_src_event_func_t_stct</code> (Structure that holds the application callback for context-level source events) . . . . .	124
<code>lbm_context_stats_t_stct</code> (Structure that holds statistics for a context) . . . .	125
<code>lbm_ctx_umq_queue_topic_list_info_t</code> (Struct containing an array of queue topics retrieved via <code>lbm_umq_queue_topic_list</code> ) . . . . .	129
<code>lbm_delete_cb_info_t_stct</code> (Structure passed to the <code>lbm_hypertopic_rcv_delete()</code> function so that a deletion callback may be called) . . . . .	130
<code>lbm_event_queue_cancel_cb_info_t_stct</code> (Structure passed to cancel/delete functions so that a cancel callback may be called) . . . . .	131

<code>lbm_event_queue_stats_t_stct</code> (Structure that holds statistics for an event queue) . . . . .	132
<code>lbm_flight_size_inflight_t_stct</code> (Structure that holds information for source total inflight messages and bytes) . . . . .	144
<code>lbm_hf_sequence_number_t_stct</code> (Structure to hold a hot failover sequence number) . . . . .	145
<code>lbm_iovec_t_stct</code> (Structure, struct iovec compatible, that holds information about buffers used for vectored sends) . . . . .	146
<code>lbm_ipv4_address_mask_t_stct</code> (Structure that holds an IPv4 address and a CIDR style netmask) . . . . .	147
<code>lbm_mim_unrecloss_func_t_stct</code> (Structure that holds the application callback for multicast immediate message unrecoverable loss notification) . . . . .	148
<code>lbm_msg_channel_info_t_stct</code> (Structure that represents UMS Spectrum channel information) . . . . .	149
<code>lbm_msg_fragment_info_t_stct</code> (Structure that holds fragment information for UM messages when appropriate) . . . . .	150
<code>lbm_msg_gateway_info_t_stct</code> (Structure that holds originating information for UM messages which arrived via a gateway) . . . . .	151
<code>lbm_msg_properties_iter_t_stct</code> (A struct used for iterating over properties pointed to by an <code>lbm_msg_properties_t</code> ) . . . . .	152
<code>lbm_msg_t_stct</code> (Structure that stores information about a received message) . . . . .	153
<code>lbm_msg_ume_deregistration_ex_t_stct</code> (Structure that holds store deregistration information for the UM receiver in an extended form) . . . . .	158
<code>lbm_msg_ume_registration_complete_ex_t_stct</code> (Structure that holds information for receivers after registration is complete to all involved stores) . . . . .	160
<code>lbm_msg_ume_registration_ex_t_stct</code> (Structure that holds store registration information for the UM receiver in an extended form) . . . . .	161
<code>lbm_msg_ume_registration_t_stct</code> (Structure that holds store registration information for the UMP receiver) . . . . .	163
<code>lbm_msg_umq_deregistration_complete_ex_t_stct</code> (Structure that holds information for receivers after they de-register from a queue) . . . . .	164
<code>lbm_msg_umq_index_assigned_ex_t_stct</code> (Structure that holds beginning-of-index information for receivers) . . . . .	165
<code>lbm_msg_umq_index_assignment_eligibility_start_complete_ex_t_stct</code> (Structure that holds index assignment information for receivers) . . . . .	166
<code>lbm_msg_umq_index_assignment_eligibility_stop_complete_ex_t_stct</code> (Structure that holds index assignment information for receivers) . . . . .	167
<code>lbm_msg_umq_index_released_ex_t_stct</code> (Structure that holds end-of-index information for receivers) . . . . .	168
<code>lbm_msg_umq_registration_complete_ex_t_stct</code> (Structure that holds information for receivers after registration is complete to all involved queue instances) . . . . .	169
<code>lbm_rcv_src_notification_func_t_stct</code> (Structure that holds the application callback for source status notifications for receivers) . . . . .	171

<code>lbm_rcv_transport_stats_daemon_t_stct</code> (Structure that holds statistics for receiver daemon mode transport (deprecated)) . . . . .	172
<code>lbm_rcv_transport_stats_lbtiipc_t_stct</code> (Structure that holds datagram statistics for receiver LBT-IPC transports) . . . . .	173
<code>lbm_rcv_transport_stats_lbtrdma_t_stct</code> (Structure that holds datagram statistics for receiver LBT-RDMA transports) . . . . .	175
<code>lbm_rcv_transport_stats_lbtrm_t_stct</code> (Structure that holds datagram statistics for receiver LBT-RM transports) . . . . .	177
<code>lbm_rcv_transport_stats_lbtru_t_stct</code> (Structure that holds datagram statistics for receiver LBT-RU transports) . . . . .	184
<code>lbm_rcv_transport_stats_t_stct</code> (Structure that holds statistics for receiver transports) . . . . .	190
<code>lbm_rcv_transport_stats_tcp_t_stct</code> (Structure that holds datagram statistics for receiver TCP transports) . . . . .	192
<code>lbm_rcv_umq_queue_msg_list_info_t</code> (Struct containing an array of UMQ messages listed via <code>lbm_rcv_umq_queue_msg_list</code> ) . . . . .	194
<code>lbm_rcv_umq_queue_msg_retrieve_info_t</code> (Struct containing an array of UMQ messages retrieved via <code>lbm_rcv_umq_queue_msg_retrieve</code> ) .	195
<code>lbm_serialized_response_t_stct</code> (Structure that holds a serialized UM response object) . . . . .	196
<code>lbm_src_cost_func_t_stct</code> (Structure that holds the "source_cost_evaluation_function" context attribute) . . . . .	197
<code>lbm_src_event_flight_size_notification_t_stct</code> (Structure that holds flight size notification event data) . . . . .	198
<code>lbm_src_event_sequence_number_info_t_stct</code> (Structure that holds sequence number information for a message sent by a source) . . . . .	199
<code>lbm_src_event_ume_ack_ex_info_t_stct</code> (Structure that holds ACK information for a given message in an extended form) . . . . .	201
<code>lbm_src_event_ume_ack_info_t_stct</code> (Structure that holds ACK information for a given message) . . . . .	203
<code>lbm_src_event_ume_deregistration_ex_t_stct</code> (Structure that holds store deregistration information for the UMP source in an extended form) .	204
<code>lbm_src_event_ume_registration_complete_ex_t_stct</code> (Structure that holds information for sources after registration is complete to all involved stores) . . . . .	206
<code>lbm_src_event_ume_registration_ex_t_stct</code> (Structure that holds store registration information for the UMP source in an extended form) . . . . .	207
<code>lbm_src_event_ume_registration_t_stct</code> (Structure that holds store registration information for the UMP source) . . . . .	209
<code>lbm_src_event_umq_message_id_info_t_stct</code> (Structure that holds Message ID information for a message sent by a sending UMQ application) .	210
<code>lbm_src_event_umq_registration_complete_ex_t_stct</code> (Structure that holds information for sources after registration is complete to all involved queue instances) . . . . .	212
<code>lbm_src_event_umq_stability_ack_info_ex_t_stct</code> (Structure that holds UMQ ACK information for a given message in an extended form) .	213

<code>lbm_src_event_umq_ulb_message_info_ex_t_stct</code> (Structure that holds UMQ ULB message information in an extended form) . . . . .	215
<code>lbm_src_event_umq_ulb_receiver_info_ex_t_stct</code> (Structure that holds UMQ ULB receiver information in an extended form) . . . . .	217
<code>lbm_src_event_wakeup_t_stct</code> (Structure that holds source wakeup event data) . . . . .	219
<code>lbm_src_notify_func_t_stct</code> (Structure that holds the callback for source notifications) . . . . .	220
<code>lbm_src_send_ex_info_t_stct</code> (Structure that holds information for the extended send calls A structure used with UM sources that utilize the extended send calls to pass options) . . . . .	221
<code>lbm_src_transport_stats_daemon_t_stct</code> (Structure that holds statistics for source daemon mode transport (deprecated)) . . . . .	223
<code>lbm_src_transport_stats_lbtpc_t_stct</code> (Structure that holds datagram statistics for source LBT-IPC transports) . . . . .	224
<code>lbm_src_transport_stats_lbtrdma_t_stct</code> (Structure that holds datagram statistics for source LBT-RDMA transports) . . . . .	225
<code>lbm_src_transport_stats_lbtrm_t_stct</code> (Structure that holds datagram statistics for source LBT-RM transports) . . . . .	226
<code>lbm_src_transport_stats_lbtru_t_stct</code> (Structure that holds datagram statistics for source LBT-RU transports) . . . . .	230
<code>lbm_src_transport_stats_t_stct</code> (Structure that holds statistics for source transports) . . . . .	233
<code>lbm_src_transport_stats_tcp_t_stct</code> (Structure that holds datagram statistics for source TCP transports) . . . . .	235
<code>lbm_str_hash_func_ex_t_stct</code> (Structure that holds the hash function callback information) . . . . .	236
<code>lbm_timeval_t_stct</code> (Structure that holds seconds and microseconds since midnight, Jan 1, 1970 UTC) . . . . .	237
<code>lbm_transport_source_info_t_stct</code> (Structure that holds formatted and parsed transport source strings) . . . . .	238
<code>lbm_uchast_resolver_entry_t_stct</code> (Structure that holds information for a unicast resolver daemon for configuration purposes) . . . . .	241
<code>lbm_ume_ctx_rcv_ctx_notification_func_t_stct</code> (Structure that holds the application callback for receiving context status notifications for source context) . . . . .	243
<code>lbm_ume_rcv_recovery_info_ex_func_info_t_stct</code> (Structure that holds information for UMP receiver recovery sequence number info application callbacks) . . . . .	244
<code>lbm_ume_rcv_recovery_info_ex_func_t_stct</code> (Structure that holds the application callback for recovery sequence number information, extended form) . . . . .	246
<code>lbm_ume_rcv_regid_ex_func_info_t_stct</code> (Structure that holds information for UMP receiver registration ID application callbacks) . . . . .	247
<code>lbm_ume_rcv_regid_ex_func_t_stct</code> (Structure that holds the application callback for registration ID setting, extended form) . . . . .	249

<code>lbm_ume_rcv_regid_func_t_stct</code> (Structure that holds the application call-back for registration ID setting ) . . . . .	250
<code>lbm_ume_src_force_reclaim_func_t_stct</code> (Structure that holds the application callback for forced reclamation notifications ) . . . . .	251
<code>lbm_ume_store_entry_t_stct</code> (Structure that holds information for a UMP store for configuration purposes ) . . . . .	252
<code>lbm_ume_store_group_entry_t_stct</code> (Structure that holds information for a UMP store group for configuration purposes ) . . . . .	253
<code>lbm_ume_store_name_entry_t_stct</code> (Structure that holds information for a UMP store by name for configuration purposes ) . . . . .	254
<code>lbm_umm_info_t_stct</code> (Structure for specifying UMM daemon connection options ) . . . . .	255
<code>lbm_umq_index_info_t_stct</code> (Structure that holds information used for sending and receiving messages with UMQ indices ) . . . . .	257
<code>lbm_umq_msg_total_lifetime_info_t_stct</code> (Structure that holds UMQ message total lifetime information ) . . . . .	258
<code>lbm_umqmsgid_t_stct</code> (Structure that holds information for UMQ messages that allows the message to be identified uniquely ) . . . . .	259
<code>lbm_umq_queue_entry_t_stct</code> (Structure that holds information for a UMQ queue registration ID for configuration purposes ) . . . . .	260
<code>lbm_umq_queue_msg_status_t</code> (Struct containing extended asynchronous operation status information about a single UMQ message ) . . . . .	261
<code>lbm_umq_queue_topic_status_t</code> (Struct containing extended asynchronous operation status information about a single UMQ topic ) . . . . .	263
<code>lbm_umq_queue_topic_t_stct</code> (Structure that holds queue topic information and can be used as a handle to a queue topic ) . . . . .	264
<code>lbm_umq_ulb_application_set_attr_t_stct</code> (Structure that holds information for a UMQ ULB sources application set attributes ) . . . . .	265
<code>lbm_umq_ulb_receiver_type_attr_t_stct</code> (Structure that holds information for a UMQ ULB sources receiver type attributes ) . . . . .	266
<code>lbm_umq_ulb_receiver_type_entry_t_stct</code> (Structure that holds information for a UMQ ULB sources receiver type associations with application sets ) . . . . .	267
<code>lbm_wildcard_rcv_compare_func_t_stct</code> (Structure that holds the application callback pattern type information for wildcard receivers ) . . . . .	268
<code>lbm_wildcard_rcv_create_func_t_stct</code> (Structure that holds the receiver creation callback information for wildcard receivers ) . . . . .	269
<code>lbm_wildcard_rcv_delete_func_t_stct</code> (Structure that holds the receiver deletion callback information for wildcard receivers ) . . . . .	270
<code>lbmmmon_attr_block_t_stct</code> (Statistics attribute block layout. Associated with each statistics message is a set of optional attributes. Any attributes present will immediately follow the packet header ) . . . . .	271
<code>lbmmmon_attr_entry_t_stct</code> (Statistics attribute entry layout. Each attribute entry within the attributes block consists of an entry header, followed immediately by the attribute data ) . . . . .	272

<code>lbmon_ctx_statistics_func_t_stct</code> (A structure that holds the callback information for context statistics ) . . . . .	273
<code>lbmon_evq_statistics_func_t_stct</code> (A structure that holds the callback information for event queue statistics ) . . . . .	274
<code>lbmon_format_func_t_stct</code> (Format module function pointer container) . .	275
<code>lbmon_packet_hdr_t_stct</code> (Statistics packet header layout) . . . . .	277
<code>lbmon_rcv_statistics_func_t_stct</code> (A structure that holds the callback information for receiver statistics) . . . . .	279
<code>lbmon_src_statistics_func_t_stct</code> (A structure that holds the callback information for source statistics) . . . . .	280
<code>lbmon_transport_func_t_stct</code> (Transport module function pointer container)	281
<code>lbmpdm_decimal_t</code> (Structure to hold a scaled decimal number. A scaled decimal number consists of a mantissa $m$ and an exponent $exp$ . It represents the value $m \cdot 10^{exp}$ ) . . . . .	283
<code>lbmpdm_field_info_attr_stct_t</code> (Attribute struct to be passed along with the name when adding field info to a definition) . . . . .	284
<code>lbmpdm_field_value_stct_t</code> (Field value struct that can be populated with a field value when passed to the <code>lbmpdm_msg_get_field_value_stct</code> function) . . . . .	285
<code>lbmpdm_timestamp_t</code> (Structure to hold a timestamp value) . . . . .	287
<code>lbmsdm_decimal_t_stct</code> (Structure to hold a scaled decimal number. A scaled decimal number consists of a mantissa $m$ and an exponent $exp$ . It represents the value $m \cdot 10^{exp}$ ) . . . . .	288
<code>ume_block_src_t_stct</code> (Structure used to designate an UME Block source) .	289
<code>ume_liveness_receiving_context_t_stct</code> (Structure that holds the information about a receiving context) . . . . .	290

# Chapter 4

## LBM API File Index

### 4.1 LBM API File List

Here is a list of all documented files with brief descriptions:

<a href="#">lbt.h</a> (Ultra Messaging (UM) API ) . . . . .	<a href="#">291</a>
<a href="#">lbmaux.h</a> (Ultra Messaging (UM) Auxiliary Functions API ) . . . . .	<a href="#">532</a>
<a href="#">lbtmht.h</a> (Ultra Messaging (UM) HyperTopic API ) . . . . .	<a href="#">536</a>
<a href="#">lbmmmon.h</a> (Ultra Messaging (UM) Monitoring API ) . . . . .	<a href="#">542</a>
<a href="#">lbmpdm.h</a> (Ultra Messaging (UM) Pre-Defined Message (PDM) API ) . . . . .	<a href="#">574</a>
<a href="#">lbmsdm.h</a> (Ultra Messaging (UM) Self-Describing Message (SDM) API ) . . . . .	<a href="#">614</a>
<a href="#">umeblocksr.h</a> (UME Blocking API ) . . . . .	<a href="#">670</a>



# **Chapter 5**

## **LBM API Page Index**

### **5.1 LBM API Related Pages**

Here is a list of all related documentation pages:

LBMMON Example source code . . . . .	<a href="#">675</a>
Deprecated List . . . . .	<a href="#">792</a>



# Chapter 6

## LBM API Module Documentation

### 6.1 Add a field to a message

#### Functions

- LBMSDMExpDLL int `lbmsdm_msg_add_boolean` (`lbmsdm_msg_t` \*Message,  
const char \*Name, uint8\_t Value)  
*Add a field to a message.*
- LBMSDMExpDLL int `lbmsdm_msg_add_int8` (`lbmsdm_msg_t` \*Message,  
const char \*Name, int8\_t Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_uint8` (`lbmsdm_msg_t` \*Message,  
const char \*Name, uint8\_t Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_int16` (`lbmsdm_msg_t` \*Message,  
const char \*Name, int16\_t Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_uint16` (`lbmsdm_msg_t` \*Message,  
const char \*Name, uint16\_t Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_int32` (`lbmsdm_msg_t` \*Message,  
const char \*Name, int32\_t Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_uint32` (`lbmsdm_msg_t` \*Message,  
const char \*Name, uint32\_t Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_int64` (`lbmsdm_msg_t` \*Message,  
const char \*Name, int64\_t Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_uint64` (`lbmsdm_msg_t` \*Message,  
const char \*Name, uint64\_t Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_float` (`lbmsdm_msg_t` \*Message,  
const char \*Name, float Value)

- LBMSDMExpDLL int `lbmsdm_msg_add_double` (`lbmsdm_msg_t` \*Message, const char \*Name, double Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_decimal` (`lbmsdm_msg_t` \*Message, const char \*Name, const `lbmsdm_decimal_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_timestamp` (`lbmsdm_msg_t` \*Message, const char \*Name, const struct timeval \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_message` (`lbmsdm_msg_t` \*Message, const char \*Name, const `lbmsdm_msg_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_string` (`lbmsdm_msg_t` \*Message, const char \*Name, const char \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_unicode` (`lbmsdm_msg_t` \*Message, const char \*Name, const wchar\_t \*Value, size\_t Length)

*Add a unicode field to a message.*

- LBMSDMExpDLL int `lbmsdm_msg_add_blob` (`lbmsdm_msg_t` \*Message, const char \*Name, const void \*Value, size\_t Length)

*Add a BLOB field to a message.*

### 6.1.1 Detailed Description

The functions in this group allow scalar (non-array) fields to be added to a message. The field value is also specified.

### 6.1.2 Function Documentation

#### 6.1.2.1 LBMSDMExpDLL int `lbmsdm_msg_add_blob` (`lbmsdm_msg_t` \**Message*, const char \**Name*, const void \**Value*, size\_t *Length*)

##### Parameters:

*Message* The SDM message to which the field is to be added.

*Name* Name of the field to be added.

*Value* Value of the field to be added.

*Length* Length of the data, in bytes.

##### Returns:

`LBMSDM_SUCCESS` if successful, `LBMSDM_FAILURE` otherwise.

**6.1.2.2 LBMSDMExpDLL int lbmsdm\_msg\_add\_boolean (lbmsdm\_msg\_t \*  
Message, const char \* Name, uint8\_t Value)**

**Parameters:**

*Message* The SDM message to which the field is to be added.

*Name* Name of the field to be added.

*Value* Value of the field to be added.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

**6.1.2.3 LBMSDMExpDLL int lbmsdm\_msg\_add\_decimal (lbmsdm\_msg\_t \*  
Message, const char \* Name, const lbmsdm\_decimal\_t \* Value)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.1.2.4 LBMSDMExpDLL int lbmsdm\_msg\_add\_double (lbmsdm\_msg\_t \*  
Message, const char \* Name, double Value)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.1.2.5 LBMSDMExpDLL int lbmsdm\_msg\_add\_float (lbmsdm\_msg\_t \*  
Message, const char \* Name, float Value)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.1.2.6 LBMSDMExpDLL int lbmsdm\_msg\_add\_int16 (lbmsdm\_msg\_t \*  
Message, const char \* Name, int16\_t Value)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.1.2.7 LBMSDMExpDLL int lbmsdm\_msg\_add\_int32 (lbmsdm\_msg\_t \*  
Message, const char \* Name, int32\_t Value)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.1.2.8 LBMSDMExpDLL int lbmsdm\_msg\_add\_int64 (lbmsdm\_msg\_t \*  
Message, const char \* Name, int64\_t Value)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.1.2.9 LBMSDMExpDLL int lbmsdm\_msg\_add\_int8 (lbmsdm\_msg\_t \*  
Message, const char \* Name, int8\_t Value)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.1.2.10 LBMSDMExpDLL int lbmsdm\_msg\_add\_message (lbmsdm\_msg\_t \*  
Message, const char \* Name, const lbmsdm\_msg\_t \* Value)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.1.2.11 LBMSDMExpDLL int lbmsdm\_msg\_add\_string (lbmsdm\_msg\_t \*  
Message, const char \* Name, const char \* Value)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.1.2.12 LBMSDMExpDLL int lbmsdm\_msg\_add\_timestamp (lbmsdm\_msg\_t \*  
\* Message, const char \* Name, const struct timeval \* Value)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.1.2.13 LBMSDMExpDLL int lbmsdm\_msg\_add\_uint16 (lbmsdm\_msg\_t \*  
Message, const char \* Name, uint16\_t Value)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.1.2.14 LBMSDMExpDLL int lbmsdm\_msg\_add\_uint32 (lbmsdm\_msg\_t \*  
Message, const char \* Name, uint32\_t Value)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.1.2.15 LBMSDMExpDLL int lbmsdm\_msg\_add\_uint64 (lbmsdm\_msg\_t \*  
*Message*, const char \* *Name*, uint64\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.1.2.16 LBMSDMExpDLL int lbmsdm\_msg\_add\_uint8 (lbmsdm\_msg\_t \*  
*Message*, const char \* *Name*, uint8\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.1.2.17 LBMSDMExpDLL int lbmsdm\_msg\_add\_unicode (lbmsdm\_msg\_t \*  
*Message*, const char \* *Name*, const wchar\_t \* *Value*, size\_t *Length*)****Parameters:**

*Message* The SDM message to which the field is to be added.

*Name* Name of the field to be added.

*Value* Value of the field to be added.

*Length* Length of the unicode string, in wchar\_ts.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

## 6.2 Add an array field to a message

### Functions

- LBMSDMExpDLL int [lbmsdm\\_msg\\_add\\_boolean\\_array](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)  
*Add an array field to a message.*
- LBMSDMExpDLL int [lbmsdm\\_msg\\_add\\_int8\\_array](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_add\\_uint8\\_array](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_add\\_int16\\_array](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_add\\_uint16\\_array](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_add\\_int32\\_array](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_add\\_uint32\\_array](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_add\\_int64\\_array](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_add\\_uint64\\_array](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_add\\_float\\_array](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_add\\_double\\_array](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_add\\_decimal\\_array](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_add\\_timestamp\\_array](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_add\\_message\\_array](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_add\\_string\\_array](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_add\\_unicode\\_array](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_add\\_blob\\_array](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)

#### 6.2.1 Detailed Description

The functions in this group allow array fields to be added to a message.

### 6.2.2 Function Documentation

#### 6.2.2.1 LBMSDMExpDLL int lbmsdm\_msg\_add\_blob\_array (lbmsdm\_msg\_t \* *Message*, const char \* *Name*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

#### 6.2.2.2 LBMSDMExpDLL int lbmsdm\_msg\_add\_boolean\_array (lbmsdm\_msg\_t \* *Message*, const char \* *Name*)

##### Parameters:

*Message* The SDM message to which the field is to be added.

*Name* Name of the field to be added.

##### Returns:

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

#### 6.2.2.3 LBMSDMExpDLL int lbmsdm\_msg\_add\_decimal\_array (lbmsdm\_msg\_t \* *Message*, const char \* *Name*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

#### 6.2.2.4 LBMSDMExpDLL int lbmsdm\_msg\_add\_double\_array (lbmsdm\_msg\_t \* *Message*, const char \* *Name*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

#### 6.2.2.5 LBMSDMExpDLL int lbmsdm\_msg\_add\_float\_array (lbmsdm\_msg\_t \* *Message*, const char \* *Name*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

#### 6.2.2.6 LBMSDMExpDLL int lbmsdm\_msg\_add\_int16\_array (lbmsdm\_msg\_t \* *Message*, const char \* *Name*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.2.2.7 LBMSDMExpDLL int lbmsdm\_msg\_add\_int32\_array (lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.2.2.8 LBMSDMExpDLL int lbmsdm\_msg\_add\_int64\_array (lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.2.2.9 LBMSDMExpDLL int lbmsdm\_msg\_add\_int8\_array (lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.2.2.10 LBMSDMExpDLL int lbmsdm\_msg\_add\_message\_array (lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.2.2.11 LBMSDMExpDLL int lbmsdm\_msg\_add\_string\_array (lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.2.2.12 LBMSDMExpDLL int lbmsdm\_msg\_add\_timestamp\_array (lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.2.2.13 LBMSDMExpDLL int lbmsdm\_msg\_add\_uint16\_array (lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.2.2.14 LBMSDMExpDLL int lbmsdm\_msg\_add\_uint32\_array  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.2.2.15 LBMSDMExpDLL int lbmsdm\_msg\_add\_uint64\_array  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.2.2.16 LBMSDMExpDLL int lbmsdm\_msg\_add\_uint8\_array  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.2.2.17 LBMSDMExpDLL int lbmsdm\_msg\_add\_unicode\_array  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

## 6.3 Add an element to an array field by field index

### Functions

- LBMSDMExpDLL int `lbmsdm_msg_add_boolean_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `uint8_t` Value)
 

*Set the value of an array field element in a message by field index.*
- LBMSDMExpDLL int `lbmsdm_msg_add_int8_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `int8_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_uint8_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `uint8_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_int16_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `int16_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_uint16_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `uint16_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_int32_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `int32_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_uint32_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `uint32_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_int64_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `int64_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_uint64_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `uint64_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_float_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, float Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_double_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, double Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_decimal_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, const `lbmsdm_decimal_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_timestamp_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, const struct timeval \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_message_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, const `lbmsdm_msg_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_string_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, const char \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_unicode_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, const wchar\_t \*Value, `size_t` Length)
 

*Set the value of a unicode array field element in a message by field index.*
- LBMSDMExpDLL int `lbmsdm_msg_add_blob_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, const void \*Value, `size_t` Length)
 

*Set the value of a blob array field element in a message by field index.*

### 6.3.1 Detailed Description

The functions in this group allow an element to be added to an array field referenced by field index.

### 6.3.2 Function Documentation

**6.3.2.1 LBMSDMExpDLL int lbmsdm\_msg\_add\_blob\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, const void \* *Value*, size\_t *Length*)**

**Parameters:**

*Message* The SDM message containing the field.

*Index* Field index.

*Value* Element value.

*Length* Length of the BLOB value, in bytes.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

**6.3.2.2 LBMSDMExpDLL int lbmsdm\_msg\_add\_boolean\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, uint8\_t *Value*)**

**Parameters:**

*Message* The SDM message containing the field.

*Index* Field index.

*Value* Element value.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

**6.3.2.3 LBMSDMExpDLL int lbmsdm\_msg\_add\_decimal\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, const lbmsdm\_decimal\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.3.2.4 LBMSDMExpDLL int lbmsdm\_msg\_add\_double\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, double *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.3.2.5 LBMSDMExpDLL int lbmsdm\_msg\_add\_float\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, float *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.3.2.6 LBMSDMExpDLL int lbmsdm\_msg\_add\_int16\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, int16\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.3.2.7 LBMSDMExpDLL int lbmsdm\_msg\_add\_int32\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, int32\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.3.2.8 LBMSDMExpDLL int lbmsdm\_msg\_add\_int64\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, int64\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.3.2.9 LBMSDMExpDLL int lbmsdm\_msg\_add\_int8\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, int8\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.3.2.10 LBMSDMExpDLL int lbmsdm\_msg\_add\_message\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, const lbmsdm\_msg\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.3.2.11 LBMSDMExpDLL int lbmsdm\_msg\_add\_string\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, const char \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.3.2.12 LBMSDMExpDLL int lbmsdm\_msg\_add\_timestamp\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, const struct timeval \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.3.2.13 LBMSDMExpDLL int lbmsdm\_msg\_add\_uint16\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, uint16\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.3.2.14 LBMSDMExpDLL int lbmsdm\_msg\_add\_uint32\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, uint32\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.3.2.15 LBMSDMExpDLL int lbmsdm\_msg\_add\_uint64\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, uint64\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.3.2.16 LBMSDMExpDLL int lbmsdm\_msg\_add\_uint8\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, uint8\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.3.2.17 LBMSDMExpDLL int lbmsdm\_msg\_add\_unicode\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, const wchar\_t \* *Value*, size\_t  
*Length*)**

**Parameters:**

*Message* The SDM message containing the field.

*Index* Field index.

*Value* Element value.

*Length* Length of the unicode string, in wchar\_ts.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

## 6.4 Add an element to an array field by field name

### Functions

- LBMSDMExpDLL int `lbmsdm_msg_add_boolean_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `uint8_t` Value)  
*Add an array field element in a message by field name.*
  - LBMSDMExpDLL int `lbmsdm_msg_add_int8_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `int8_t` Value)
  - LBMSDMExpDLL int `lbmsdm_msg_add_uint8_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `uint8_t` Value)
  - LBMSDMExpDLL int `lbmsdm_msg_add_int16_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `int16_t` Value)
  - LBMSDMExpDLL int `lbmsdm_msg_add_uint16_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `uint16_t` Value)
  - LBMSDMExpDLL int `lbmsdm_msg_add_int32_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `int32_t` Value)
  - LBMSDMExpDLL int `lbmsdm_msg_add_uint32_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `uint32_t` Value)
  - LBMSDMExpDLL int `lbmsdm_msg_add_int64_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `int64_t` Value)
  - LBMSDMExpDLL int `lbmsdm_msg_add_uint64_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `uint64_t` Value)
  - LBMSDMExpDLL int `lbmsdm_msg_add_float_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, float Value)
  - LBMSDMExpDLL int `lbmsdm_msg_add_double_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, double Value)
  - LBMSDMExpDLL int `lbmsdm_msg_add_decimal_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, const `lbmsdm_decimal_t` \*Value)
  - LBMSDMExpDLL int `lbmsdm_msg_add_timestamp_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, const struct timeval \*Value)
  - LBMSDMExpDLL int `lbmsdm_msg_add_message_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, const `lbmsdm_msg_t` \*Value)
  - LBMSDMExpDLL int `lbmsdm_msg_add_string_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, const char \*Value)
  - LBMSDMExpDLL int `lbmsdm_msg_add_unicode_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, const `wchar_t` \*Value, `size_t` Length)  
*Add a unicode array field element in a message by field name.*
  - LBMSDMExpDLL int `lbmsdm_msg_add_blob_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, const void \*Value, `size_t` Length)  
*Add a BLOB array field element in a message by field name.*

### 6.4.1 Detailed Description

The functions in this group allow an element to be added to an array field referenced by field name.

### 6.4.2 Function Documentation

#### 6.4.2.1 LBMSDMExpDLL int lbmsdm\_msg\_add\_blob\_elem\_name (lbmsdm\_msg\_t \* *Message*, const char \* *Name*, const void \* *Value*, size\_t *Length*)

##### Parameters:

*Message* The SDM message containing the field.

*Name* Field name.

*Value* Element value.

*Length* Length of the BLOB data, in bytes.

##### Returns:

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

#### 6.4.2.2 LBMSDMExpDLL int lbmsdm\_msg\_add\_boolean\_elem\_name (lbmsdm\_msg\_t \* *Message*, const char \* *Name*, uint8\_t *Value*)

##### Parameters:

*Message* The SDM message containing the field.

*Name* Field name.

*Value* Element value.

##### Returns:

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

#### 6.4.2.3 LBMSDMExpDLL int lbmsdm\_msg\_add\_decimal\_elem\_name (lbmsdm\_msg\_t \* *Message*, const char \* *Name*, const lbmsdm\_decimal\_t \* *Value*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.4.2.4 LBMSDMExpDLL int lbmsdm\_msg\_add\_double\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, double *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.4.2.5 LBMSDMExpDLL int lbmsdm\_msg\_add\_float\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, float *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.4.2.6 LBMSDMExpDLL int lbmsdm\_msg\_add\_int16\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, int16\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.4.2.7 LBMSDMExpDLL int lbmsdm\_msg\_add\_int32\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, int32\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.4.2.8 LBMSDMExpDLL int lbmsdm\_msg\_add\_int64\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, int64\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.4.2.9 LBMSDMExpDLL int lbmsdm\_msg\_add\_int8\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, int8\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.4.2.10 LBMSDMExpDLL int lbmsdm\_msg\_add\_message\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, const lbmsdm\_msg\_t \*  
*Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.4.2.11 LBMSDMExpDLL int lbmsdm\_msg\_add\_string\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, const char \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.4.2.12 LBMSDMExpDLL int lbmsdm\_msg\_add\_timestamp\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, const struct timeval \*  
*Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.4.2.13 LBMSDMExpDLL int lbmsdm\_msg\_add\_uint16\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, uint16\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.4.2.14 LBMSDMExpDLL int lbmsdm\_msg\_add\_uint32\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, uint32\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.4.2.15 LBMSDMExpDLL int lbmsdm\_msg\_add\_uint64\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, uint64\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.4.2.16 LBMSDMExpDLL int lbmsdm\_msg\_add\_uint8\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, uint8\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.4.2.17 LBMSDMExpDLL int lbmsdm\_msg\_add\_unicode\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, const wchar\_t \* *Value*,  
size\_t *Length*)**

**Parameters:**

*Message* The SDM message containing the field.

*Name* Field name.

*Value* Element value.

*Length* Length of the unicode string, in wchar\_ts.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

## 6.5 Add an element to an array field referenced by an iterator

### Functions

- LBMSDMExpDLL int [lbmsdm\\_iter\\_add\\_boolean\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, [uint8\\_t](#) Value)

*Add an array field element in a message referenced by an iterator.*

- LBMSDMExpDLL int [lbmsdm\\_iter\\_add\\_int8\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, [int8\\_t](#) Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_add\\_uint8\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, [uint8\\_t](#) Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_add\\_int16\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, [int16\\_t](#) Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_add\\_uint16\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, [uint16\\_t](#) Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_add\\_int32\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, [int32\\_t](#) Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_add\\_uint32\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, [uint32\\_t](#) Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_add\\_int64\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, [int64\\_t](#) Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_add\\_uint64\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, [uint64\\_t](#) Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_add\\_float\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, [float](#) Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_add\\_double\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, [double](#) Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_add\\_decimal\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, const [lbmsdm\\_decimal\\_t](#) \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_add\\_timestamp\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, const struct timeval \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_add\\_message\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, const [lbmsdm\\_msg\\_t](#) \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_add\\_string\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, const char \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_add\\_unicode\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, const wchar\_t \*Value, size\_t Length)

*Add a unicode array field element in a message referenced by an iterator.*

- LBMSDMExpDLL int [lbmsdm\\_iter\\_add\\_blob\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, const void \*Value, size\_t Length)

Add a BLOB array field element in a message referenced by an iterator.

### 6.5.1 Detailed Description

The functions in this group allow an element to be added to an array field referenced by an iterator.

### 6.5.2 Function Documentation

#### 6.5.2.1 LBMSDMExpDLL int lbmsdm\_iter\_add\_blob\_elem (lbmsdm\_iter\_t \* \* *Iterator*, const void \* *Value*, size\_t *Length*)

##### Parameters:

*Iterator* The iterator referencing the field.

*Value* Element value.

*Length* Length of the BLOB data, in bytes.

##### Returns:

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

#### 6.5.2.2 LBMSDMExpDLL int lbmsdm\_iter\_add\_boolean\_elem (lbmsdm\_iter\_t \* *Iterator*, uint8\_t *Value*)

##### Parameters:

*Iterator* The iterator referencing the field.

*Value* Element value.

##### Returns:

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

#### 6.5.2.3 LBMSDMExpDLL int lbmsdm\_iter\_add\_decimal\_elem (lbmsdm\_iter\_t \* *Iterator*, const lbmsdm\_decimal\_t \* *Value*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.5.2.4 LBMSDMExpDLL int lbmsdm\_iter\_add\_double\_elem (lbmsdm\_iter\_t \* *Iterator*, double *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.5.2.5 LBMSDMExpDLL int lbmsdm\_iter\_add\_float\_elem (lbmsdm\_iter\_t \* *Iterator*, float *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.5.2.6 LBMSDMExpDLL int lbmsdm\_iter\_add\_int16\_elem (lbmsdm\_iter\_t \* *Iterator*, int16\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.5.2.7 LBMSDMExpDLL int lbmsdm\_iter\_add\_int32\_elem (lbmsdm\_iter\_t \* *Iterator*, int32\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.5.2.8 LBMSDMExpDLL int lbmsdm\_iter\_add\_int64\_elem (lbmsdm\_iter\_t \* *Iterator*, int64\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.5.2.9 LBMSDMExpDLL int lbmsdm\_iter\_add\_int8\_elem (lbmsdm\_iter\_t \* *Iterator*, int8\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.5.2.10 LBMSDMExpDLL int lbmsdm\_iter\_add\_message\_elem (lbmsdm\_iter\_t \* *Iterator*, const lbmsdm\_msg\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.5.2.11 LBMSDMExpDLL int lbmsdm\_iter\_add\_string\_elem (lbmsdm\_iter\_t \* *Iterator*, const char \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.5.2.12 LBMSDMExpDLL int lbmsdm\_iter\_add\_timestamp\_elem (lbmsdm\_iter\_t \* *Iterator*, const struct timeval \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.5.2.13 LBMSDMExpDLL int lbmsdm\_iter\_add\_uint16\_elem (lbmsdm\_iter\_t \* *Iterator*, uint16\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.5.2.14 LBMSDMExpDLL int lbmsdm\_iter\_add\_uint32\_elem (lbmsdm\_iter\_t \* *Iterator*, uint32\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.5.2.15 LBMSDMExpDLL int lbmsdm\_iter\_add\_uint64\_elem (lbmsdm\_iter\_t \* *Iterator*, uint64\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.5.2.16 LBMSDMExpDLL int lbmsdm\_iter\_add\_uint8\_elem (lbmsdm\_iter\_t \* *Iterator*, uint8\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.5.2.17 LBMSDMExpDLL int lbmsdm\_iter\_add\_unicode\_elem (lbmsdm\_iter\_t \* *Iterator*, const wchar\_t \* *Value*, size\_t *Length*)****Parameters:**

*Iterator* The iterator referencing the field.

*Value* Element value.

*Length* Length of the unicode string, in wchar\_ts.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

## 6.6 Get scalar field values by field index

### Functions

- LBMSDMExpDLL int [`lbmsdm\_msg\_get\_boolean\_idx`](#) ([`lbmsdm\_msg\_t`](#) \*Message, size\_t Index, uint8\_t \*Value)

*Fetch a field value from a message by field index.*
- LBMSDMExpDLL int [`lbmsdm\_msg\_get\_int8\_idx`](#) ([`lbmsdm\_msg\_t`](#) \*Message, size\_t Index, int8\_t \*Value)
- LBMSDMExpDLL int [`lbmsdm\_msg\_get\_uint8\_idx`](#) ([`lbmsdm\_msg\_t`](#) \*Message, size\_t Index, uint8\_t \*Value)
- LBMSDMExpDLL int [`lbmsdm\_msg\_get\_int16\_idx`](#) ([`lbmsdm\_msg\_t`](#) \*Message, size\_t Index, int16\_t \*Value)
- LBMSDMExpDLL int [`lbmsdm\_msg\_get\_uint16\_idx`](#) ([`lbmsdm\_msg\_t`](#) \*Message, size\_t Index, uint16\_t \*Value)
- LBMSDMExpDLL int [`lbmsdm\_msg\_get\_int32\_idx`](#) ([`lbmsdm\_msg\_t`](#) \*Message, size\_t Index, int32\_t \*Value)
- LBMSDMExpDLL int [`lbmsdm\_msg\_get\_uint32\_idx`](#) ([`lbmsdm\_msg\_t`](#) \*Message, size\_t Index, uint32\_t \*Value)
- LBMSDMExpDLL int [`lbmsdm\_msg\_get\_int64\_idx`](#) ([`lbmsdm\_msg\_t`](#) \*Message, size\_t Index, int64\_t \*Value)
- LBMSDMExpDLL int [`lbmsdm\_msg\_get\_uint64\_idx`](#) ([`lbmsdm\_msg\_t`](#) \*Message, size\_t Index, uint64\_t \*Value)
- LBMSDMExpDLL int [`lbmsdm\_msg\_get\_float\_idx`](#) ([`lbmsdm\_msg\_t`](#) \*Message, size\_t Index, float \*Value)
- LBMSDMExpDLL int [`lbmsdm\_msg\_get\_double\_idx`](#) ([`lbmsdm\_msg\_t`](#) \*Message, size\_t Index, double \*Value)
- LBMSDMExpDLL int [`lbmsdm\_msg\_get\_decimal\_idx`](#) ([`lbmsdm\_msg\_t`](#) \*Message, size\_t Index, [`lbmsdm\_decimal\_t`](#) \*Value)
- LBMSDMExpDLL int [`lbmsdm\_msg\_get\_timestamp\_idx`](#) ([`lbmsdm\_msg\_t`](#) \*Message, size\_t Index, struct timeval \*Value)
- LBMSDMExpDLL int [`lbmsdm\_msg\_get\_message\_idx`](#) ([`lbmsdm\_msg\_t`](#) \*Message, size\_t Index, [`lbmsdm\_msg\_t`](#) \*\*Value)
- LBMSDMExpDLL int [`lbmsdm\_msg\_get\_string\_idx`](#) ([`lbmsdm\_msg\_t`](#) \*Message, size\_t Index, char \*Value, size\_t \*Size)

*Fetch a string field value from a message by field index.*
- LBMSDMExpDLL int [`lbmsdm\_msg\_get\_unicode\_idx`](#) ([`lbmsdm\_msg\_t`](#) \*Message, size\_t Index, wchar\_t \*Value, size\_t \*Size)

*Fetch a unicode field value from a message by field index.*
- LBMSDMExpDLL int [`lbmsdm\_msg\_get\_blob\_idx`](#) ([`lbmsdm\_msg\_t`](#) \*Message, size\_t Index, void \*Value, size\_t \*Size)

Fetch a BLOB field value from a message by field index.

### 6.6.1 Detailed Description

The functions in this group allow the retrieval of the value of a scalar (non-array) field, referenced by field index.

### 6.6.2 Function Documentation

#### 6.6.2.1 LBMSDMExpDLL int lbmsdm\_msg\_get\_blob\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, void \* *Value*, size\_t \* *Size*)

##### Parameters:

*Message* The SDM message from which the field is to be fetched.

*Index* Field index.

*Value* Pointer to variable where the value is stored.

*Size* Pointer to a variable containing the maximum size of *Value* in bytes. On exit, it will contain the actual size of *Value*.

##### Return values:

**LBMSDM\_SUCCESS** if successful

**LBMSDM\_INSUFFICIENT\_BUFFER\_LENGTH** if *Size* is not large enough for the data. *Size* will contain the length required.

**LBMSDM\_FAILURE** otherwise.

#### 6.6.2.2 LBMSDMExpDLL int lbmsdm\_msg\_get\_boolean\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, uint8\_t \* *Value*)

##### Parameters:

*Message* The SDM message from which the field is to be fetched.

*Index* Field index.

*Value* Pointer to variable where the value is stored.

##### Returns:

**LBMSDM\_SUCCESS** if successful, **LBMSDM\_FAILURE** otherwise.

**6.6.2.3 LBMSDMExpDLL int lbmsdm\_msg\_get\_decimal\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, lbmsdm\_decimal\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.6.2.4 LBMSDMExpDLL int lbmsdm\_msg\_get\_double\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, double \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.6.2.5 LBMSDMExpDLL int lbmsdm\_msg\_get\_float\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, float \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.6.2.6 LBMSDMExpDLL int lbmsdm\_msg\_get\_int16\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, int16\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.6.2.7 LBMSDMExpDLL int lbmsdm\_msg\_get\_int32\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, int32\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.6.2.8 LBMSDMExpDLL int lbmsdm\_msg\_get\_int64\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, int64\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.6.2.9 LBMSDMExpDLL int lbmsdm\_msg\_get\_int8\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, int8\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.6.2.10 LBMSDMExpDLL int lbmsdm\_msg\_get\_message\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, lbmsdm\_msg\_t \*\* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.6.2.11 LBMSDMExpDLL int lbmsdm\_msg\_get\_string\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, char \* *Value*, size\_t \* *Size*)**

**Parameters:**

***Message*** The SDM message from which the field is to be fetched.

***Index*** Field index.

***Value*** Pointer to variable where the value is stored.

***Size*** Pointer to a variable containing the maximum size of *Value* (including the terminating null character). On exit, it will contain the actual size of the string (including the terminating null character).

**Return values:**

***LBMSDM\_SUCCESS*** if successful

***LBMSDM\_INSUFFICIENT\_BUFFER\_LENGTH*** if *Size* is not large enough for the string. *Size* will contain the length required.

***LBMSDM\_FAILURE*** otherwise.

**6.6.2.12 LBMSDMExpDLL int lbmsdm\_msg\_get\_timestamp\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, struct timeval \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.6.2.13 LBMSDMExpDLL int lbmsdm\_msg\_get\_uint16\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, uint16\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.6.2.14 LBMSDMExpDLL int lbmsdm\_msg\_get\_uint32\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, uint32\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.6.2.15 LBMSDMExpDLL int lbmsdm\_msg\_get\_uint64\_idx (lbmsdm\_msg\_t \*  
*Message*, size\_t *Index*, uint64\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.6.2.16 LBMSDMExpDLL int lbmsdm\_msg\_get\_uint8\_idx (lbmsdm\_msg\_t \*  
*Message*, size\_t *Index*, uint8\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.6.2.17 LBMSDMExpDLL int lbmsdm\_msg\_get\_unicode\_idx (lbmsdm\_msg\_t  
\* *Message*, size\_t *Index*, wchar\_t \* *Value*, size\_t \* *Size*)****Parameters:**

*Message* The SDM message from which the field is to be fetched.

*Index* Field index.

*Value* Pointer to variable where the value is stored.

*Size* Pointer to a variable containing the maximum size of *Value* in wchar\_ts.  
On exit, it will contain the actual size of *Value* in wchar\_ts.

**Return values:**

***LBMSDM\_SUCCESS*** if successful

***LBMSDM\_INSUFFICIENT\_BUFFER\_LENGTH*** if *Size* is not large enough  
for the data. *Size* will contain the length required.

***LBMSDM\_FAILURE*** otherwise.

## 6.7 Get scalar field values by field name

### Functions

- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_boolean\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, uint8\_t \*Value)
 

*Fetch a field value from a message by field name.*
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_int8\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, int8\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_uint8\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, uint8\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_int16\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, int16\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_uint16\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, uint16\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_int32\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, int32\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_uint32\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, uint32\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_int64\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, int64\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_uint64\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, uint64\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_float\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, float \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_double\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, double \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_decimal\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, lbmsdm\_decimal\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_timestamp\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, struct timeval \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_message\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, lbmsdm\_msg\_t \*\*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_string\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, char \*Value, size\_t \*Size)
 

*Fetch a string field value from a message by field name.*
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_unicode\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, wchar\_t \*Value, size\_t \*Size)
 

*Fetch a unicode field value from a message by field name.*
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_blob\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, void \*Value, size\_t \*Size)

Fetch a BLOB field value from a message by field name.

### 6.7.1 Detailed Description

The functions in this group allow the retrieval of the value of a scalar (non-array) field, referenced by field name.

### 6.7.2 Function Documentation

#### 6.7.2.1 LBMSDMExpDLL int lbmsdm\_msg\_get\_blob\_name (lbmsdm\_msg\_t \* *Message*, const char \* *Name*, void \* *Value*, size\_t \* *Size*)

##### Parameters:

*Message* The SDM message from which the field is to be fetched.

*Name* Field name.

*Value* Pointer to variable where the value is stored.

*Size* Pointer to a variable containing the maximum size of *Value* in bytes. On exit, it will contain the actual size of the data.

##### Return values:

**LBMSDM\_SUCCESS** if successful

**LBMSDM\_INSUFFICIENT\_BUFFER\_LENGTH** if *Size* is not large enough for the string. *Size* will contain the length required in bytes.

**LBMSDM\_FAILURE** otherwise.

#### 6.7.2.2 LBMSDMExpDLL int lbmsdm\_msg\_get\_boolean\_name (lbmsdm\_msg\_t \* *Message*, const char \* *Name*, uint8\_t \* *Value*)

##### Parameters:

*Message* The SDM message from which the field is to be fetched.

*Name* Field name.

*Value* Pointer to variable where the value is stored.

##### Returns:

**LBMSDM\_SUCCESS** if successful, **LBMSDM\_FAILURE** otherwise.

**6.7.2.3 LBMSDMExpDLL int lbmsdm\_msg\_get\_decimal\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, lbmsdm\_decimal\_t \*  
*Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.7.2.4 LBMSDMExpDLL int lbmsdm\_msg\_get\_double\_name (lbmsdm\_msg\_t  
\* *Message*, const char \* *Name*, double \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.7.2.5 LBMSDMExpDLL int lbmsdm\_msg\_get\_float\_name (lbmsdm\_msg\_t \*  
*Message*, const char \* *Name*, float \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.7.2.6 LBMSDMExpDLL int lbmsdm\_msg\_get\_int16\_name (lbmsdm\_msg\_t \*  
*Message*, const char \* *Name*, int16\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.7.2.7 LBMSDMExpDLL int lbmsdm\_msg\_get\_int32\_name (lbmsdm\_msg\_t \*  
*Message*, const char \* *Name*, int32\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.7.2.8 LBMSDMExpDLL int lbmsdm\_msg\_get\_int64\_name (lbmsdm\_msg\_t \*  
*Message*, const char \* *Name*, int64\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.7.2.9 LBMSDMExpDLL int lbmsdm\_msg\_get\_int8\_name (lbmsdm\_msg\_t \*  
*Message*, const char \* *Name*, int8\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.7.2.10 LBMSDMExpDLL int lbmsdm\_msg\_get\_message\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, lbmsdm\_msg\_t \*\*  
*Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.7.2.11 LBMSDMExpDLL int lbmsdm\_msg\_get\_string\_name (lbmsdm\_msg\_t  
\* *Message*, const char \* *Name*, char \* *Value*, size\_t \* *Size*)**

**Parameters:**

*Message* The SDM message from which the field is to be fetched.

*Name* Field name.

*Value* Pointer to variable where the value is stored.

*Size* Pointer to a variable containing the maximum size of *Value* (including the terminating null character). On exit, it will contain the actual size of the string (including the terminating null character).

**Return values:**

***LBMSDM\_SUCCESS*** if successful

***LBMSDM\_INSUFFICIENT\_BUFFER\_LENGTH*** if *Size* is not large enough for the string. *Size* will contain the length required.

***LBMSDM\_FAILURE*** otherwise.

**6.7.2.12 LBMSDMExpDLL int lbmsdm\_msg\_get\_timestamp\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, struct timeval \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.7.2.13 LBMSDMExpDLL int lbmsdm\_msg\_get\_uint16\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, uint16\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.7.2.14 LBMSDMExpDLL int lbmsdm\_msg\_get\_uint32\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, uint32\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.7.2.15 LBMSDMExpDLL int lbmsdm\_msg\_get\_uint64\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, uint64\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.7.2.16 LBMSDMExpDLL int lbmsdm\_msg\_get\_uint8\_name (lbmsdm\_msg\_t  
\* *Message*, const char \* *Name*, uint8\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.7.2.17 LBMSDMExpDLL int lbmsdm\_msg\_get\_unicode\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, wchar\_t \* *Value*, size\_t  
\* *Size*)**

**Parameters:**

*Message* The SDM message from which the field is to be fetched.

*Name* Field name.

*Value* Pointer to variable where the value is stored.

*Size* Pointer to a variable containing the maximum size of *Value* in wchar\_ts.  
On exit, it will contain the actual size of the data in wchar\_ts

**Return values:**

***LBMSDM\_SUCCESS*** if successful

***LBMSDM\_INSUFFICIENT\_BUFFER\_LENGTH*** if *Size* is not large enough  
for the string. *Size* will contain the length required in wchar\_ts.

***LBMSDM\_FAILURE*** otherwise.

## 6.8 Get a scalar field via an iterator

### Functions

- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_boolean](#) (lbmsdm\_iter\_t \*Iterator, uint8\_t \*Value)

*Fetch a field value from the field referenced by an iterator.*

- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_int8](#) (lbmsdm\_iter\_t \*Iterator, int8\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_uint8](#) (lbmsdm\_iter\_t \*Iterator, uint8\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_int16](#) (lbmsdm\_iter\_t \*Iterator, int16\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_uint16](#) (lbmsdm\_iter\_t \*Iterator, uint16\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_int32](#) (lbmsdm\_iter\_t \*Iterator, int32\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_uint32](#) (lbmsdm\_iter\_t \*Iterator, uint32\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_int64](#) (lbmsdm\_iter\_t \*Iterator, int64\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_uint64](#) (lbmsdm\_iter\_t \*Iterator, uint64\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_float](#) (lbmsdm\_iter\_t \*Iterator, float \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_double](#) (lbmsdm\_iter\_t \*Iterator, double \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_decimal](#) (lbmsdm\_iter\_t \*Iterator, lbmsdm\_decimal\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_timestamp](#) (lbmsdm\_iter\_t \*Iterator, struct timeval \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_message](#) (lbmsdm\_iter\_t \*Iterator, lbmsdm\_msg\_t \*\*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_string](#) (lbmsdm\_iter\_t \*Iterator, char \*Value, size\_t \*Size)

*Fetch a string field value from the field referenced by an iterator.*

- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_unicode](#) (lbmsdm\_iter\_t \*Iterator, wchar\_t \*Value, size\_t \*Size)

*Fetch a unicode field value from the field referenced by an iterator.*

- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_blob](#) (lbmsdm\_iter\_t \*Iterator, void \*Value, size\_t \*Size)

*Fetch a BLOB field value from the field referenced by an iterator.*

### 6.8.1 Detailed Description

The functions in this group allow the retrieval of the value of a scalar (non-array) field, referenced by an iterator.

### 6.8.2 Function Documentation

#### 6.8.2.1 LBMSDMExpDLL int lbmsdm\_iter\_get\_blob (lbmsdm\_iter\_t \* *Iterator*, void \* *Value*, size\_t \* *Size*)

**Parameters:**

*Iterator* The SDM iterator to use.

*Value* Pointer to variable where the value is stored.

*Size* Pointer to a variable containing the maximum size of *Value* in bytes. On exit, it will contain the actual size of the data.

**Return values:**

[LBMSDM\\_SUCCESS](#) if successful

[LBMSDM\\_INSUFFICIENT\\_BUFFER\\_LENGTH](#) if *Size* is not large enough for the data. *Size* will contain the length required in bytes.

[LBMSDM\\_FAILURE](#) otherwise.

#### 6.8.2.2 LBMSDMExpDLL int lbmsdm\_iter\_get\_boolean (lbmsdm\_iter\_t \* *Iterator*, uint8\_t \* *Value*)

**Parameters:**

*Iterator* The SDM iterator to use.

*Value* Pointer to variable where the value is stored.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

#### 6.8.2.3 LBMSDMExpDLL int lbmsdm\_iter\_get\_decimal (lbmsdm\_iter\_t \* *Iterator*, lbmsdm\_decimal\_t \* *Value*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.8.2.4 LBMSDMExpDLL int lbmsdm\_iter\_get\_double (lbmsdm\_iter\_t \*  
*Iterator*, double \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.8.2.5 LBMSDMExpDLL int lbmsdm\_iter\_get\_float (lbmsdm\_iter\_t \*  
*Iterator*, float \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.8.2.6 LBMSDMExpDLL int lbmsdm\_iter\_get\_int16 (lbmsdm\_iter\_t \*  
*Iterator*, int16\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.8.2.7 LBMSDMExpDLL int lbmsdm\_iter\_get\_int32 (lbmsdm\_iter\_t \*  
*Iterator*, int32\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.8.2.8 LBMSDMExpDLL int lbmsdm\_iter\_get\_int64 (lbmsdm\_iter\_t \*  
*Iterator*, int64\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.8.2.9 LBMSDMExpDLL int lbmsdm\_iter\_get\_int8 (lbmsdm\_iter\_t \* *Iterator*,  
int8\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.8.2.10 LBMSDMExpDLL int lbmsdm\_iter\_get\_message (lbmsdm\_iter\_t \*  
*Iterator*, lbmsdm\_msg\_t \*\* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.8.2.11 LBMSDMExpDLL int lbmsdm\_iter\_get\_string (lbmsdm\_iter\_t \*  
*Iterator*, char \* *Value*, size\_t \* *Size*)**

**Parameters:**

***Iterator*** The SDM iterator to use.

***Value*** Pointer to variable where the value is stored.

***Size*** Pointer to a variable containing the maximum size of *Value* (including the terminating null character). On exit, it will contain the actual size of the string (including the terminating null character).

**Return values:**

***LBMSDM\_SUCCESS*** if successful

***LBMSDM\_INSUFFICIENT\_BUFFER\_LENGTH*** if *Size* is not large enough for the string. *Size* will contain the length required.

***LBMSDM\_FAILURE*** otherwise.

**6.8.2.12 LBMSDMExpDLL int lbmsdm\_iter\_get\_timestamp (lbmsdm\_iter\_t \*  
*Iterator*, struct timeval \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.8.2.13 LBMSDMExpDLL int lbmsdm\_iter\_get\_uint16 (lbmsdm\_iter\_t \*  
*Iterator*, uint16\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.8.2.14 LBMSDMExpDLL int lbmsdm\_iter\_get\_uint32 (lbmsdm\_iter\_t \*  
*Iterator*, uint32\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.8.2.15 LBMSDMExpDLL int lbmsdm\_iter\_get\_uint64 (lbmsdm\_iter\_t \*  
*Iterator*, uint64\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.8.2.16 LBMSDMExpDLL int lbmsdm\_iter\_get\_uint8 (lbmsdm\_iter\_t \*  
Iterator, uint8\_t \* Value)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.8.2.17 LBMSDMExpDLL int lbmsdm\_iter\_get\_unicode (lbmsdm\_iter\_t \*  
Iterator, wchar\_t \* Value, size\_t \* Size)****Parameters:**

**Iterator** The SDM iterator to use.

**Value** Pointer to variable where the value is stored.

**Size** Pointer to a variable containing the maximum size of **Value** in wchar\_ts.  
On exit, it will contain the actual size of the data in wchar\_ts.

**Return values:**

**LBMSDM\_SUCCESS** if successful

**LBMSDM\_INSUFFICIENT\_BUFFER\_LENGTH** if **Size** is not large enough  
for the data. **Size** will contain the length required in wchar\_ts.

**LBMSDM\_FAILURE** otherwise.

## 6.9 Get an element from an array field by field index

### Functions

- LBMSDMExpDLL int `lbmsdm_msg_get_boolean_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, uint8\_t \*Value)

*Fetch an array field element value from a message by field index.*

- LBMSDMExpDLL int `lbmsdm_msg_get_int8_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, int8\_t \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_uint8_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, uint8\_t \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_int16_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, int16\_t \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_uint16_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, uint16\_t \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_int32_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, int32\_t \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_uint32_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, uint32\_t \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_int64_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, int64\_t \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_uint64_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, uint64\_t \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_float_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, float \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_double_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, double \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_decimal_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, `lbmsdm_decimal_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_timestamp_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, struct timeval \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_message_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, `lbmsdm_msg_t` \*\*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_string_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, char \*Value, size\_t \*Size)

*Fetch a string array field element value from a message by field index.*

- LBMSDMExpDLL int `lbmsdm_msg_get_unicode_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, wchar\_t \*Value, size\_t \*Size)

*Fetch a unicode array field element value from a message by field index.*

- LBMSDMExpDLL int `lbmsdm_msg_get_blob_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, void \*Value, size\_t \*Size)

Fetch a BLOB array field element value from a message by field index.

### 6.9.1 Detailed Description

The functions in this group allow the retrieval of an element value of an array field, referenced by field index.

### 6.9.2 Function Documentation

**6.9.2.1 LBMSDMExpDLL int lbmsdm\_msg\_get\_blob\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, void \* *Value*,  
size\_t \* *Size*)**

**Parameters:**

***Message*** The SDM message from which the field is to be fetched.

***Index*** Field index.

***Element*** Element number (zero-based).

***Value*** Pointer to variable where the value is stored.

***Size*** Pointer to a variable containing the maximum size of *Value* in bytes.. On exit, it will contain the actual size of the data.

**Return values:**

***LBMSDM\_SUCCESS*** if successful

***LBMSDM\_INSUFFICIENT\_BUFFER\_LENGTH*** if *Size* is not large enough for the data. *Size* will contain the length required in bytes.

***LBMSDM\_FAILURE*** otherwise.

**6.9.2.2 LBMSDMExpDLL int lbmsdm\_msg\_get\_boolean\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, uint8\_t \* *Value*)**

**Parameters:**

***Message*** The SDM message from which the field is to be fetched.

***Index*** Field index.

***Element*** Element number (zero-based).

***Value*** Pointer to variable where the value is stored.

**Returns:**

***LBMSDM\_SUCCESS*** if successful, ***LBMSDM\_FAILURE*** otherwise.

**6.9.2.3 LBMSDMExpDLL int lbmsdm\_msg\_get\_decimal\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*,  
lbmsdm\_decimal\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.9.2.4 LBMSDMExpDLL int lbmsdm\_msg\_get\_double\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, double \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.9.2.5 LBMSDMExpDLL int lbmsdm\_msg\_get\_float\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, float \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.9.2.6 LBMSDMExpDLL int lbmsdm\_msg\_get\_int16\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, int16\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.9.2.7 LBMSDMExpDLL int lbmsdm\_msg\_get\_int32\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, int32\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.9.2.8 LBMSDMExpDLL int lbmsdm\_msg\_get\_int64\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, int64\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.9.2.9 LBMSDMExpDLL int lbmsdm\_msg\_get\_int8\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, int8\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.9.2.10 LBMSDMExpDLL int lbmsdm\_msg\_get\_message\_elem\_idx**  
(*lbmsdm\_msg\_t \* Message*, *size\_t Index*, *size\_t Element*,  
*lbmsdm\_msg\_t \*\* Value*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.9.2.11 LBMSDMExpDLL int lbmsdm\_msg\_get\_string\_elem\_idx**  
(*lbmsdm\_msg\_t \* Message*, *size\_t Index*, *size\_t Element*, *char \* Value*,  
*size\_t \* Size*)

**Parameters:**

*Message* The SDM message from which the field is to be fetched.

*Index* Field index.

*Element* Element number (zero-based).

*Value* Pointer to variable where the value is stored.

*Size* Pointer to a variable containing the maximum size of *Value* (including the terminating null character). On exit, it will contain the actual size of the string (including the terminating null character).

**Return values:**

***LBMSDM\_SUCCESS*** if successful

***LBMSDM\_INSUFFICIENT\_BUFFER\_LENGTH*** if *Size* is not large enough for the string. *Size* will contain the length required.

***LBMSDM\_FAILURE*** otherwise.

**6.9.2.12 LBMSDMExpDLL int lbmsdm\_msg\_get\_timestamp\_elem\_idx**  
(*lbmsdm\_msg\_t \* Message*, *size\_t Index*, *size\_t Element*, *struct timeval*  
*\* Value*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.9.2.13 LBMSDMExpDLL int lbmsdm\_msg\_get\_uint16\_elem\_idx**  
(*lbmsdm\_msg\_t \* Message*, *size\_t Index*, *size\_t Element*, *uint16\_t*  
*\* Value*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

---

**6.9.2.14 LBMSDMExpDLL int lbmsdm\_msg\_get\_uint32\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, uint32\_t \*  
*Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.9.2.15 LBMSDMExpDLL int lbmsdm\_msg\_get\_uint64\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, uint64\_t \*  
*Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.9.2.16 LBMSDMExpDLL int lbmsdm\_msg\_get\_uint8\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, uint8\_t \*  
*Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.9.2.17 LBMSDMExpDLL int lbmsdm\_msg\_get\_unicode\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, wchar\_t \*  
*Value*, size\_t \* *Size*)**

**Parameters:**

*Message* The SDM message from which the field is to be fetched.

*Index* Field index.

*Element* Element number (zero-based).

*Value* Pointer to variable where the value is stored.

*Size* Pointer to a variable containing the maximum size of *Value* in wchar\_ts.  
On exit, it will contain the actual size of the data in wchar\_ts.

**Return values:**

***LBMSDM\_SUCCESS*** if successful

***LBMSDM\_INSUFFICIENT\_BUFFER\_LENGTH*** if *Size* is not large enough  
for the data. *Size* will contain the length required in wchar\_ts.

***LBMSDM\_FAILURE*** otherwise.

## 6.10 Get an element from an array field by field name

### Functions

- LBMSDMExpDLL int `lbmsdm_msg_get_boolean_elem_name` (`lbmsdm_msg_t *Message, const char *Name, size_t Element, uint8_t *Value`)  
*Fetch an array field element value from a message by field name.*
  - LBMSDMExpDLL int `lbmsdm_msg_get_int8_elem_name` (`lbmsdm_msg_t *Message, const char *Name, size_t Element, int8_t *Value`)
  - LBMSDMExpDLL int `lbmsdm_msg_get_uint8_elem_name` (`lbmsdm_msg_t *Message, const char *Name, size_t Element, uint8_t *Value`)
  - LBMSDMExpDLL int `lbmsdm_msg_get_int16_elem_name` (`lbmsdm_msg_t *Message, const char *Name, size_t Element, int16_t *Value`)
  - LBMSDMExpDLL int `lbmsdm_msg_get_uint16_elem_name` (`lbmsdm_msg_t *Message, const char *Name, size_t Element, uint16_t *Value`)
  - LBMSDMExpDLL int `lbmsdm_msg_get_int32_elem_name` (`lbmsdm_msg_t *Message, const char *Name, size_t Element, int32_t *Value`)
  - LBMSDMExpDLL int `lbmsdm_msg_get_uint32_elem_name` (`lbmsdm_msg_t *Message, const char *Name, size_t Element, uint32_t *Value`)
  - LBMSDMExpDLL int `lbmsdm_msg_get_int64_elem_name` (`lbmsdm_msg_t *Message, const char *Name, size_t Element, int64_t *Value`)
  - LBMSDMExpDLL int `lbmsdm_msg_get_uint64_elem_name` (`lbmsdm_msg_t *Message, const char *Name, size_t Element, uint64_t *Value`)
  - LBMSDMExpDLL int `lbmsdm_msg_get_float_elem_name` (`lbmsdm_msg_t *Message, const char *Name, size_t Element, float *Value`)
  - LBMSDMExpDLL int `lbmsdm_msg_get_double_elem_name` (`lbmsdm_msg_t *Message, const char *Name, size_t Element, double *Value`)
  - LBMSDMExpDLL int `lbmsdm_msg_get_decimal_elem_name` (`lbmsdm_msg_t *Message, const char *Name, size_t Element, lbmsdm_decimal_t *Value`)
  - LBMSDMExpDLL int `lbmsdm_msg_get_timestamp_elem_name` (`lbmsdm_msg_t *Message, const char *Name, size_t Element, struct timeval *Value`)
  - LBMSDMExpDLL int `lbmsdm_msg_get_message_elem_name` (`lbmsdm_msg_t *Message, const char *Name, size_t Element, lbmsdm_msg_t **Value`)
  - LBMSDMExpDLL int `lbmsdm_msg_get_string_elem_name` (`lbmsdm_msg_t *Message, const char *Name, size_t Element, char *Value, size_t *Size`)  
*Fetch a string array field element value from a message by field name.*
  - LBMSDMExpDLL int `lbmsdm_msg_get_unicode_elem_name` (`lbmsdm_msg_t *Message, const char *Name, size_t Element, wchar_t *Value, size_t *Size`)  
*Fetch a unicode array field element value from a message by field name.*
  - LBMSDMExpDLL int `lbmsdm_msg_get_blob_elem_name` (`lbmsdm_msg_t *Message, const char *Name, size_t Element, void *Value, size_t *Size`)

---

*Fetch a BLOB array field element value from a message by field name.*

### 6.10.1 Detailed Description

The functions in this group allow the retrieval of an element value of an array field, referenced by field name.

### 6.10.2 Function Documentation

#### 6.10.2.1 LBMSDMExpDLL int lbmsdm\_msg\_get\_blob\_elem\_name (lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*, void \*   *Value*, size\_t \* *Size*)

**Parameters:**

*Message* The SDM message from which the field is to be fetched.

*Name* Field name.

*Element* Element number (zero-based).

*Value* Pointer to variable where the value is stored.

*Size* Pointer to a variable containing the maximum size of *Value* in bytes. On exit, it will contain the actual size of the data.

**Return values:**

***LBMSDM\_SUCCESS*** if successful

***LBMSDM\_INSUFFICIENT\_BUFFER\_LENGTH*** if *Size* is not large enough for the data. *Size* will contain the length required in bytes.

***LBMSDM\_FAILURE*** otherwise.

#### 6.10.2.2 LBMSDMExpDLL int lbmsdm\_msg\_get\_boolean\_elem\_name (lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*, uint8\_t   \* *Value*)

**Parameters:**

*Message* The SDM message from which the field is to be fetched.

*Name* Field name.

*Element* Element number (zero-based).

*Value* Pointer to variable where the value is stored.

**Returns:**

***LBMSDM\_SUCCESS*** if successful, ***LBMSDM\_FAILURE*** otherwise.

**6.10.2.3 LBMSDMExpDLL int lbmsdm\_msg\_get\_decimal\_elem\_name**  
(*lbmsdm\_msg\_t \* Message*, *const char \* Name*, *size\_t Element*,  
*lbmsdm\_decimal\_t \* Value*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.10.2.4 LBMSDMExpDLL int lbmsdm\_msg\_get\_double\_elem\_name**  
(*lbmsdm\_msg\_t \* Message*, *const char \* Name*, *size\_t Element*, *double \* Value*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.10.2.5 LBMSDMExpDLL int lbmsdm\_msg\_get\_float\_elem\_name**  
(*lbmsdm\_msg\_t \* Message*, *const char \* Name*, *size\_t Element*, *float \* Value*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.10.2.6 LBMSDMExpDLL int lbmsdm\_msg\_get\_int16\_elem\_name**  
(*lbmsdm\_msg\_t \* Message*, *const char \* Name*, *size\_t Element*, *int16\_t \* Value*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.10.2.7 LBMSDMExpDLL int lbmsdm\_msg\_get\_int32\_elem\_name**  
(*lbmsdm\_msg\_t \* Message*, *const char \* Name*, *size\_t Element*, *int32\_t \* Value*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.10.2.8 LBMSDMExpDLL int lbmsdm\_msg\_get\_int64\_elem\_name**  
(*lbmsdm\_msg\_t \* Message*, *const char \* Name*, *size\_t Element*, *int64\_t \* Value*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.10.2.9 LBMSDMDExpDLL int lbmsdm\_msg\_get\_int8\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*, int8\_t \*  
*Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.10.2.10 LBMSDMDExpDLL int lbmsdm\_msg\_get\_message\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*,  
lbmsdm\_msg\_t \*\* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.10.2.11 LBMSDMDExpDLL int lbmsdm\_msg\_get\_string\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*, char \*  
*Value*, size\_t \* *Size*)**

**Parameters:**

*Message* The SDM message from which the field is to be fetched.

*Name* Field name.

*Element* Element number (zero-based).

*Value* Pointer to variable where the value is stored.

*Size* Pointer to a variable containing the maximum size of *Value* (including the terminating null character). On exit, it will contain the actual size of the string (including the terminating null character).

**Return values:**

***LBMSDM\_SUCCESS*** if successful

***LBMSDM\_INSUFFICIENT\_BUFFER\_LENGTH*** if *Size* is not large enough for the string. *Size* will contain the length required.

***LBMSDM\_FAILURE*** otherwise.

**6.10.2.12 LBMSDMDExpDLL int lbmsdm\_msg\_get\_timestamp\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*, struct  
timeval \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.10.2.13 LBMSDMExpDLL int lbmsdm\_msg\_get\_uint16\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*,  
uint16\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.10.2.14 LBMSDMExpDLL int lbmsdm\_msg\_get\_uint32\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*,  
uint32\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.10.2.15 LBMSDMExpDLL int lbmsdm\_msg\_get\_uint64\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*,  
uint64\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.10.2.16 LBMSDMExpDLL int lbmsdm\_msg\_get\_uint8\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*,  
uint8\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.10.2.17 LBMSDMExpDLL int lbmsdm\_msg\_get\_unicode\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*,  
wchar\_t \* *Value*, size\_t \* *Size*)**

**Parameters:**

***Message*** The SDM message from which the field is to be fetched.

***Name*** Field name.

***Element*** Element number (zero-based).

***Value*** Pointer to variable where the value is stored.

***Size*** Pointer to a variable containing the maximum size of *Value* in wchar\_ts.

On exit, it will contain the actual size of the data.

**Return values:**

***LBMSDM\_SUCCESS*** if successful

***LBMSDM\_INSUFFICIENT\_BUFFER\_LENGTH*** if *Size* is not large enough for the data. *Size* will contain the length required in `wchar_ts`.

***LBMSDM\_FAILURE*** otherwise.

## 6.11 Get an element from an array field referenced by an iterator

### Functions

- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_boolean\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, uint8\_t \*Value)

*Fetch an array field element value from the field referenced by an iterator.*

- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_int8\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, int8\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_uint8\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, uint8\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_int16\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, int16\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_uint16\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, uint16\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_int32\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, int32\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_uint32\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, uint32\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_int64\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, int64\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_uint64\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, uint64\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_float\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, float \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_double\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, double \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_decimal\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, lbmsdm\_decimal\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_timestamp\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, struct timeval \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_message\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, lbmsdm\_msg\_t \*\*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_string\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, char \*Value, size\_t \*Size)

*Fetch a string array field element value from the field referenced by an iterator.*

- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_unicode\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, wchar\_t \*Value, size\_t \*Size)

*Fetch a unicode array field element value from the field referenced by an iterator.*

- LBMSDMExpDLL int `lbmsdm_iter_get_blob_elem` (`lbmsdm_iter_t *Iterator,`  
`size_t Element, void *Value, size_t *Size`)

*Fetch a blob array field element value from the field referenced by an iterator.*

### 6.11.1 Detailed Description

The functions in this group allow the retrieval of an element value of an array field, referenced by an iterator.

### 6.11.2 Function Documentation

#### 6.11.2.1 LBMSDMExpDLL int `lbmsdm_iter_get_blob_elem` (`lbmsdm_iter_t *Iterator,` `size_t Element, void * Value, size_t * Size`)

##### Parameters:

*Iterator* The SDM iterator to use.

*Element* Element number (zero-based).

*Value* Pointer to variable where the value is stored.

*Size* Pointer to a variable containing the maximum size of *Value* in bytes. On exit, it will contain the actual size of the data.

##### Return values:

`LBMSDM_SUCCESS` if successful

`LBMSDM_INSUFFICIENT_BUFFER_LENGTH` if *Size* is not large enough for the data. *Size* will contain the length required in bytes.

`LBMSDM_FAILURE` otherwise.

#### 6.11.2.2 LBMSDMExpDLL int `lbmsdm_iter_get_boolean_elem` `(lbmsdm_iter_t *Iterator, size_t Element, uint8_t * Value)`

##### Parameters:

*Iterator* The SDM iterator to use.

*Element* Element number (zero-based).

*Value* Pointer to variable where the value is stored.

##### Returns:

`LBMSDM_SUCCESS` if successful, `LBMSDM_FAILURE` otherwise.

**6.11.2.3 LBMSDMExpDLL int lbmsdm\_iter\_get\_decimal\_elem  
(lbmsdm\_iter\_t \* *Iterator*, size\_t *Element*, lbmsdm\_decimal\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.11.2.4 LBMSDMExpDLL int lbmsdm\_iter\_get\_double\_elem (lbmsdm\_iter\_t \*  
\* *Iterator*, size\_t *Element*, double \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.11.2.5 LBMSDMExpDLL int lbmsdm\_iter\_get\_float\_elem (lbmsdm\_iter\_t \*  
\* *Iterator*, size\_t *Element*, float \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.11.2.6 LBMSDMExpDLL int lbmsdm\_iter\_get\_int16\_elem (lbmsdm\_iter\_t \*  
\* *Iterator*, size\_t *Element*, int16\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.11.2.7 LBMSDMExpDLL int lbmsdm\_iter\_get\_int32\_elem (lbmsdm\_iter\_t \*  
\* *Iterator*, size\_t *Element*, int32\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.11.2.8 LBMSDMExpDLL int lbmsdm\_iter\_get\_int64\_elem (lbmsdm\_iter\_t \*  
\* *Iterator*, size\_t *Element*, int64\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.11.2.9 LBMSDMExpDLL int lbmsdm\_iter\_get\_int8\_elem (lbmsdm\_iter\_t \*  
\* *Iterator*, size\_t *Element*, int8\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

---

**6.11.2.10 LBMSDMDLL int lbmsdm\_iter\_get\_message\_elem  
(lbmsdm\_iter\_t \* *Iterator*, size\_t *Element*, lbmsdm\_msg\_t \*\* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.11.2.11 LBMSDMDLL int lbmsdm\_iter\_get\_string\_elem (lbmsdm\_iter\_t  
\* *Iterator*, size\_t *Element*, char \* *Value*, size\_t \* *Size*)**
**Parameters:**

*Iterator* The SDM iterator to use.

*Element* Element number (zero-based).

*Value* Pointer to variable where the value is stored.

*Size* Pointer to a variable containing the maximum size of *Value* (including the terminating null character). On exit, it will contain the actual size of the string (including the terminating null character).

**Return values:**

***LBMSDM\_SUCCESS*** if successful

***LBMSDM\_INSUFFICIENT\_BUFFER\_LENGTH*** if *Size* is not large enough for the string. *Size* will contain the length required.

***LBMSDM\_FAILURE*** otherwise.

**6.11.2.12 LBMSDMDLL int lbmsdm\_iter\_get\_timestamp\_elem  
(lbmsdm\_iter\_t \* *Iterator*, size\_t *Element*, struct timeval \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.11.2.13 LBMSDMDLL int lbmsdm\_iter\_get\_uint16\_elem (lbmsdm\_iter\_t  
\* *Iterator*, size\_t *Element*, uint16\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.11.2.14 LBMSDMDLL int lbmsdm\_iter\_get\_uint32\_elem (lbmsdm\_iter\_t  
\* *Iterator*, size\_t *Element*, uint32\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.11.2.15 LBMSDMExpDLL int lbmsdm\_iter\_get\_uint64\_elem (lbmsdm\_iter\_t \* *Iterator*, size\_t *Element*, uint64\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.11.2.16 LBMSDMExpDLL int lbmsdm\_iter\_get\_uint8\_elem (lbmsdm\_iter\_t \* *Iterator*, size\_t *Element*, uint8\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.11.2.17 LBMSDMExpDLL int lbmsdm\_iter\_get\_unicode\_elem  
(lbmsdm\_iter\_t \* *Iterator*, size\_t *Element*, wchar\_t \* *Value*, size\_t \* *Size*)****Parameters:**

***Iterator*** The SDM iterator to use.

***Element*** Element number (zero-based).

***Value*** Pointer to variable where the value is stored.

***Size*** Pointer to a variable containing the maximum size of *Value* in wchar\_ts.  
On exit, it will contain the actual size of the data.

**Return values:**

***LBMSDM\_SUCCESS*** if successful

***LBMSDM\_INSUFFICIENT\_BUFFER\_LENGTH*** if *Size* is not large enough  
for the data. *Size* will contain the length required in wchar\_ts.

***LBMSDM\_FAILURE*** otherwise.

## 6.12 Set a field value in a message by field index

### Functions

- LBMSDMExpDLL int `lbmsdm_msg_set_boolean_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `uint8_t` Value)
 

*Set a field value in a message by field index.*
- LBMSDMExpDLL int `lbmsdm_msg_set_int8_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `int8_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_uint8_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `uint8_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_int16_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `int16_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_uint16_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `uint16_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_int32_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `int32_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_uint32_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `uint32_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_int64_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `int64_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_uint64_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `uint64_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_float_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `float` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_double_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `double` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_decimal_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, const `lbmsdm_decimal_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_timestamp_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, const struct timeval \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_message_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, const `lbmsdm_msg_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_string_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, const char \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_unicode_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, const wchar\_t \*Value, `size_t` Length)
 

*Set a unicode field value in a message by field index.*
- LBMSDMExpDLL int `lbmsdm_msg_set_blob_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, const void \*Value, `size_t` Length)
 

*Set a BLOB field value in a message by field index.*

### 6.12.1 Detailed Description

The functions in this group allow the value of a field to be set, and the type of the field to be set to a scalar type, for a field referenced by field index.

### 6.12.2 Function Documentation

#### 6.12.2.1 LBMSDMExpDLL int lbmsdm\_msg\_set\_blob\_idx (lbmsdm\_msg\_t \* \* *Message*, size\_t *Index*, const void \* *Value*, size\_t *Length*)

##### Parameters:

*Message* The SDM message containing the field.

*Index* Field index.

*Value* Pointer to variable containing the value.

*Length* Length of *Value* in bytes.

##### Returns:

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

#### 6.12.2.2 LBMSDMExpDLL int lbmsdm\_msg\_set\_boolean\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, uint8\_t *Value*)

##### Parameters:

*Message* The SDM message containing the field.

*Index* Field index.

*Value* Pointer to variable containing the value.

##### Returns:

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

#### 6.12.2.3 LBMSDMExpDLL int lbmsdm\_msg\_set\_decimal\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, const lbmsdm\_decimal\_t \* *Value*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.12.2.4 LBMSDMExpDLL int lbmsdm\_msg\_set\_double\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, double *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.12.2.5 LBMSDMExpDLL int lbmsdm\_msg\_set\_float\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, float *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.12.2.6 LBMSDMExpDLL int lbmsdm\_msg\_set\_int16\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, int16\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.12.2.7 LBMSDMExpDLL int lbmsdm\_msg\_set\_int32\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, int32\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.12.2.8 LBMSDMExpDLL int lbmsdm\_msg\_set\_int64\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, int64\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.12.2.9 LBMSDMExpDLL int lbmsdm\_msg\_set\_int8\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, int8\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.12.2.10 LBMSDMExpDLL int lbmsdm\_msg\_set\_message\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, const lbmsdm\_msg\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.12.2.11 LBMSDMExpDLL int lbmsdm\_msg\_set\_string\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, const char \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.12.2.12 LBMSDMExpDLL int lbmsdm\_msg\_set\_timestamp\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, const struct timeval \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.12.2.13 LBMSDMExpDLL int lbmsdm\_msg\_set\_uint16\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, uint16\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.12.2.14 LBMSDMExpDLL int lbmsdm\_msg\_set\_uint32\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, uint32\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.12.2.15 LBMSDMExpDLL int lbmsdm\_msg\_set\_uint64\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, uint64\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.12.2.16 LBMSDMExpDLL int lbmsdm\_msg\_set\_uint8\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, uint8\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.12.2.17 LBMSDMDExpDLL int lbmsdm\_msg\_set\_unicode\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, const wchar\_t \* *Value*, size\_t  
*Length*)**

**Parameters:**

*Message* The SDM message containing the field.

*Index* Field index.

*Value* Pointer to variable containing the value.

*Length* Length of *Value* in wchar\_ts.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

## 6.13 Set a field value in a message by field name

### Functions

- LBMSDMExpDLL int [`lbmsdm\_msg\_set\_boolean\_name`](#) ([`lbmsdm\_msg\_t`](#) \*Message, const char \*Name, uint8\_t Value)  
*Set a field value in a message by field name.*
  - LBMSDMExpDLL int [`lbmsdm\_msg\_set\_int8\_name`](#) ([`lbmsdm\_msg\_t`](#) \*Message, const char \*Name, int8\_t Value)
  - LBMSDMExpDLL int [`lbmsdm\_msg\_set\_uint8\_name`](#) ([`lbmsdm\_msg\_t`](#) \*Message, const char \*Name, uint8\_t Value)
  - LBMSDMExpDLL int [`lbmsdm\_msg\_set\_int16\_name`](#) ([`lbmsdm\_msg\_t`](#) \*Message, const char \*Name, int16\_t Value)
  - LBMSDMExpDLL int [`lbmsdm\_msg\_set\_uint16\_name`](#) ([`lbmsdm\_msg\_t`](#) \*Message, const char \*Name, uint16\_t Value)
  - LBMSDMExpDLL int [`lbmsdm\_msg\_set\_int32\_name`](#) ([`lbmsdm\_msg\_t`](#) \*Message, const char \*Name, int32\_t Value)
  - LBMSDMExpDLL int [`lbmsdm\_msg\_set\_uint32\_name`](#) ([`lbmsdm\_msg\_t`](#) \*Message, const char \*Name, uint32\_t Value)
  - LBMSDMExpDLL int [`lbmsdm\_msg\_set\_int64\_name`](#) ([`lbmsdm\_msg\_t`](#) \*Message, const char \*Name, int64\_t Value)
  - LBMSDMExpDLL int [`lbmsdm\_msg\_set\_uint64\_name`](#) ([`lbmsdm\_msg\_t`](#) \*Message, const char \*Name, uint64\_t Value)
  - LBMSDMExpDLL int [`lbmsdm\_msg\_set\_float\_name`](#) ([`lbmsdm\_msg\_t`](#) \*Message, const char \*Name, float Value)
  - LBMSDMExpDLL int [`lbmsdm\_msg\_set\_double\_name`](#) ([`lbmsdm\_msg\_t`](#) \*Message, const char \*Name, double Value)
  - LBMSDMExpDLL int [`lbmsdm\_msg\_set\_decimal\_name`](#) ([`lbmsdm\_msg\_t`](#) \*Message, const char \*Name, const [`lbmsdm\_decimal\_t`](#) \*Value)
  - LBMSDMExpDLL int [`lbmsdm\_msg\_set\_timestamp\_name`](#) ([`lbmsdm\_msg\_t`](#) \*Message, const char \*Name, const struct timeval \*Value)
  - LBMSDMExpDLL int [`lbmsdm\_msg\_set\_message\_name`](#) ([`lbmsdm\_msg\_t`](#) \*Message, const char \*Name, const [`lbmsdm\_msg\_t`](#) \*Value)
  - LBMSDMExpDLL int [`lbmsdm\_msg\_set\_string\_name`](#) ([`lbmsdm\_msg\_t`](#) \*Message, const char \*Name, const char \*Value)
  - LBMSDMExpDLL int [`lbmsdm\_msg\_set\_unicode\_name`](#) ([`lbmsdm\_msg\_t`](#) \*Message, const char \*Name, const wchar\_t \*Value, size\_t Length)  
*Set a unicode field value in a message by field name.*
  - LBMSDMExpDLL int [`lbmsdm\_msg\_set\_blob\_name`](#) ([`lbmsdm\_msg\_t`](#) \*Message, const char \*Name, const void \*Value, size\_t Length)  
*Set a BLOB field value in a message by field name.*

### 6.13.1 Detailed Description

The functions in this group allow the value of a field to be set, and the type of the field to be set to a scalar type, for a field referenced by field name.

### 6.13.2 Function Documentation

#### 6.13.2.1 LBMSDMExpDLL int lbmsdm\_msg\_set\_blob\_name (lbmsdm\_msg\_t \* *Message*, const char \* *Name*, const void \* *Value*, size\_t *Length*)

##### Parameters:

*Message* The SDM message containing the field.

*Name* Field name.

*Value* New value.

*Length* Length of *Value* in bytes.

##### Returns:

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

#### 6.13.2.2 LBMSDMExpDLL int lbmsdm\_msg\_set\_boolean\_name (lbmsdm\_msg\_t \* *Message*, const char \* *Name*, uint8\_t *Value*)

##### Parameters:

*Message* The SDM message containing the field.

*Name* Field name.

*Value* New field value.

##### Returns:

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

#### 6.13.2.3 LBMSDMExpDLL int lbmsdm\_msg\_set\_decimal\_name (lbmsdm\_msg\_t \* *Message*, const char \* *Name*, const lbmsdm\_decimal\_t \* *Value*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.13.2.4 LBMSDMExpDLL int lbmsdm\_msg\_set\_double\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, double *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.13.2.5 LBMSDMExpDLL int lbmsdm\_msg\_set\_float\_name (lbmsdm\_msg\_t  
\* *Message*, const char \* *Name*, float *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.13.2.6 LBMSDMExpDLL int lbmsdm\_msg\_set\_int16\_name (lbmsdm\_msg\_t  
\* *Message*, const char \* *Name*, int16\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.13.2.7 LBMSDMExpDLL int lbmsdm\_msg\_set\_int32\_name (lbmsdm\_msg\_t  
\* *Message*, const char \* *Name*, int32\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.13.2.8 LBMSDMExpDLL int lbmsdm\_msg\_set\_int64\_name (lbmsdm\_msg\_t  
\* *Message*, const char \* *Name*, int64\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.13.2.9 LBMSDMExpDLL int lbmsdm\_msg\_set\_int8\_name (lbmsdm\_msg\_t \*  
*Message*, const char \* *Name*, int8\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.13.2.10 LBMSDMExpDLL int lbmsdm\_msg\_set\_message\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, const lbmsdm\_msg\_t  
\* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.13.2.11 LBMSDMDLL int lbmsdm\_msg\_set\_string\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, const char \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.13.2.12 LBMSDMDLL int lbmsdm\_msg\_set\_timestamp\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, const struct timeval \*  
*Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.13.2.13 LBMSDMDLL int lbmsdm\_msg\_set\_uint16\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, uint16\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.13.2.14 LBMSDMDLL int lbmsdm\_msg\_set\_uint32\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, uint32\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.13.2.15 LBMSDMDLL int lbmsdm\_msg\_set\_uint64\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, uint64\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.13.2.16 LBMSDMDLL int lbmsdm\_msg\_set\_uint8\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, uint8\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.13.2.17 LBMSDMExpDLL int lbmsdm\_msg\_set\_unicode\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, const wchar\_t \* *Value*,  
size\_t *Length*)**

**Parameters:**

*Message* The SDM message containing the field.

*Name* Field name.

*Value* New value.

*Length* Length of *Value* in wchar\_ts.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

## 6.14 Set a field value in a message referenced by an iterator

### Functions

- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_boolean](#) (lbmsdm\_iter\_t \*Iterator, uint8\_t Value)

*Set a field value in the field referenced by an iterator.*

- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_int8](#) (lbmsdm\_iter\_t \*Iterator, int8\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_uint8](#) (lbmsdm\_iter\_t \*Iterator, uint8\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_int16](#) (lbmsdm\_iter\_t \*Iterator, int16\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_uint16](#) (lbmsdm\_iter\_t \*Iterator, uint16\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_int32](#) (lbmsdm\_iter\_t \*Iterator, int32\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_uint32](#) (lbmsdm\_iter\_t \*Iterator, uint32\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_int64](#) (lbmsdm\_iter\_t \*Iterator, int64\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_uint64](#) (lbmsdm\_iter\_t \*Iterator, uint64\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_float](#) (lbmsdm\_iter\_t \*Iterator, float Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_double](#) (lbmsdm\_iter\_t \*Iterator, double Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_decimal](#) (lbmsdm\_iter\_t \*Iterator, const lbmsdm\_decimal\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_timestamp](#) (lbmsdm\_iter\_t \*Iterator, const struct timeval \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_message](#) (lbmsdm\_iter\_t \*Iterator, const lbmsdm\_msg\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_string](#) (lbmsdm\_iter\_t \*Iterator, const char \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_unicode](#) (lbmsdm\_iter\_t \*Iterator, const wchar\_t \*Value, size\_t Length)

*Set a unicode field value in the field referenced by an iterator.*

- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_blob](#) (lbmsdm\_iter\_t \*Iterator, const void \*Value, size\_t Length)

*Set a BLOB field value in the field referenced by an iterator.*

### 6.14.1 Detailed Description

The functions in this group allow the value of a field to be set, and the type of the field to be set to a scalar type, for a field referenced by an iterator.

### 6.14.2 Function Documentation

#### 6.14.2.1 LBMSDMExpDLL int lbmsdm\_iter\_set\_blob (lbmsdm\_iter\_t \* *Iterator*, const void \* *Value*, size\_t *Length*)

**Parameters:**

*Iterator* The SDM iterator to use.

*Value* New value.

*Length* Length of *Value* in bytes.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

#### 6.14.2.2 LBMSDMExpDLL int lbmsdm\_iter\_set\_boolean (lbmsdm\_iter\_t \* *Iterator*, uint8\_t *Value*)

**Parameters:**

*Iterator* The SDM iterator to use.

*Value* The new field value.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

#### 6.14.2.3 LBMSDMExpDLL int lbmsdm\_iter\_set\_decimal (lbmsdm\_iter\_t \* *Iterator*, const lbmsdm\_decimal\_t \* *Value*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.14.2.4 LBMSDMExpDLL int lbmsdm\_iter\_set\_double (lbmsdm\_iter\_t \*  
*Iterator*, double *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.14.2.5 LBMSDMExpDLL int lbmsdm\_iter\_set\_float (lbmsdm\_iter\_t \*  
*Iterator*, float *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.14.2.6 LBMSDMExpDLL int lbmsdm\_iter\_set\_int16 (lbmsdm\_iter\_t \*  
*Iterator*, int16\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.14.2.7 LBMSDMExpDLL int lbmsdm\_iter\_set\_int32 (lbmsdm\_iter\_t \*  
*Iterator*, int32\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.14.2.8 LBMSDMExpDLL int lbmsdm\_iter\_set\_int64 (lbmsdm\_iter\_t \*  
*Iterator*, int64\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.14.2.9 LBMSDMExpDLL int lbmsdm\_iter\_set\_int8 (lbmsdm\_iter\_t \*  
*Iterator*, int8\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.14.2.10 LBMSDMExpDLL int lbmsdm\_iter\_set\_message (lbmsdm\_iter\_t \*  
*Iterator*, const lbmsdm\_msg\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.14.2.11 LBMSDMExpDLL int lbmsdm\_iter\_set\_string (lbmsdm\_iter\_t \*  
*Iterator*, const char \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.14.2.12 LBMSDMExpDLL int lbmsdm\_iter\_set\_timestamp (lbmsdm\_iter\_t \*  
*Iterator*, const struct timeval \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.14.2.13 LBMSDMExpDLL int lbmsdm\_iter\_set\_uint16 (lbmsdm\_iter\_t \*  
*Iterator*, uint16\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.14.2.14 LBMSDMExpDLL int lbmsdm\_iter\_set\_uint32 (lbmsdm\_iter\_t \*  
*Iterator*, uint32\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.14.2.15 LBMSDMExpDLL int lbmsdm\_iter\_set\_uint64 (lbmsdm\_iter\_t \*  
*Iterator*, uint64\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.14.2.16 LBMSDMExpDLL int lbmsdm\_iter\_set\_uint8 (lbmsdm\_iter\_t \*  
*Iterator*, uint8\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.14.2.17 LBMSDMExpDLL int lbmsdm\_iter\_set\_unicode (lbmsdm\_iter\_t \*  
*Iterator*, const wchar\_t \* *Value*, size\_t *Length*)****Parameters:**

*Iterator* The SDM iterator to use.

*Value* New value.

*Length* Length of *Value* in wchar\_ts.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

## 6.15 Set a field value in a message by field index to an array field

### Functions

- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_boolean\\_array\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)  
*Set a field in a message by field index to an array field.*
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_int8\\_array\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_uint8\\_array\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_int16\\_array\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_uint16\\_array\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_int32\\_array\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_uint32\\_array\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_int64\\_array\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_uint64\\_array\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_float\\_array\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_double\\_array\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_decimal\\_array\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_timestamp\\_array\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_message\\_array\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_string\\_array\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_unicode\\_array\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_blob\\_array\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)

### 6.15.1 Detailed Description

The functions in this group allow the type of the field to be set to an array type, for a field referenced by field index.

### 6.15.2 Function Documentation

#### 6.15.2.1 LBMSDMExpDLL int lbmsdm\_msg\_set\_blob\_array\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

#### 6.15.2.2 LBMSDMExpDLL int lbmsdm\_msg\_set\_boolean\_array\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*)

**Parameters:**

*Message* The SDM message containing the field.

*Index* Field index.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

#### 6.15.2.3 LBMSDMExpDLL int lbmsdm\_msg\_set\_decimal\_array\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

#### 6.15.2.4 LBMSDMExpDLL int lbmsdm\_msg\_set\_double\_array\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

#### 6.15.2.5 LBMSDMExpDLL int lbmsdm\_msg\_set\_float\_array\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.15.2.6 LBMSDMExpDLL int lbmsdm\_msg\_set\_int16\_array\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.15.2.7 LBMSDMExpDLL int lbmsdm\_msg\_set\_int32\_array\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.15.2.8 LBMSDMExpDLL int lbmsdm\_msg\_set\_int64\_array\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.15.2.9 LBMSDMExpDLL int lbmsdm\_msg\_set\_int8\_array\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.15.2.10 LBMSDMExpDLL int lbmsdm\_msg\_set\_message\_array\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.15.2.11 LBMSDMExpDLL int lbmsdm\_msg\_set\_string\_array\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.15.2.12 LBMSDMExpDLL int lbmsdm\_msg\_set\_timestamp\_array\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.15.2.13 LBMSDMDExpDLL int lbmsdm\_msg\_set\_uint16\_array\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.15.2.14 LBMSDMDExpDLL int lbmsdm\_msg\_set\_uint32\_array\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.15.2.15 LBMSDMDExpDLL int lbmsdm\_msg\_set\_uint64\_array\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.15.2.16 LBMSDMDExpDLL int lbmsdm\_msg\_set\_uint8\_array\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.15.2.17 LBMSDMDExpDLL int lbmsdm\_msg\_set\_unicode\_array\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

## 6.16 Set a field value in a message by field name to an array field

### Functions

- LBMSDMExpDLL int `lbmsdm_msg_set_boolean_array_name` (`lbmsdm_msg_t *Message, const char *Name)`

*Set a field in a message by field name to an array field.*
- LBMSDMExpDLL int `lbmsdm_msg_set_int8_array_name` (`lbmsdm_msg_t *Message, const char *Name)`
- LBMSDMExpDLL int `lbmsdm_msg_set_uint8_array_name` (`lbmsdm_msg_t *Message, const char *Name)`
- LBMSDMExpDLL int `lbmsdm_msg_set_int16_array_name` (`lbmsdm_msg_t *Message, const char *Name)`
- LBMSDMExpDLL int `lbmsdm_msg_set_uint16_array_name` (`lbmsdm_msg_t *Message, const char *Name)`
- LBMSDMExpDLL int `lbmsdm_msg_set_int32_array_name` (`lbmsdm_msg_t *Message, const char *Name)`
- LBMSDMExpDLL int `lbmsdm_msg_set_uint32_array_name` (`lbmsdm_msg_t *Message, const char *Name)`
- LBMSDMExpDLL int `lbmsdm_msg_set_int64_array_name` (`lbmsdm_msg_t *Message, const char *Name)`
- LBMSDMExpDLL int `lbmsdm_msg_set_uint64_array_name` (`lbmsdm_msg_t *Message, const char *Name)`
- LBMSDMExpDLL int `lbmsdm_msg_set_float_array_name` (`lbmsdm_msg_t *Message, const char *Name)`
- LBMSDMExpDLL int `lbmsdm_msg_set_double_array_name` (`lbmsdm_msg_t *Message, const char *Name)`
- LBMSDMExpDLL int `lbmsdm_msg_set_decimal_array_name` (`lbmsdm_msg_t *Message, const char *Name)`
- LBMSDMExpDLL int `lbmsdm_msg_set_timestamp_array_name` (`lbmsdm_msg_t *Message, const char *Name)`
- LBMSDMExpDLL int `lbmsdm_msg_set_message_array_name` (`lbmsdm_msg_t *Message, const char *Name)`
- LBMSDMExpDLL int `lbmsdm_msg_set_string_array_name` (`lbmsdm_msg_t *Message, const char *Name)`
- LBMSDMExpDLL int `lbmsdm_msg_set_unicode_array_name` (`lbmsdm_msg_t *Message, const char *Name)`
- LBMSDMExpDLL int `lbmsdm_msg_set_blob_array_name` (`lbmsdm_msg_t *Message, const char *Name)`

### 6.16.1 Detailed Description

The functions in this group allow the type of the field to be set to an array type, for a field referenced by field name.

### 6.16.2 Function Documentation

#### 6.16.2.1 LBMSDMDLL int lbmsdm\_msg\_set\_blob\_array\_name (lbmsdm\_msg\_t \* *Message*, const char \* *Name*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

#### 6.16.2.2 LBMSDMDLL int lbmsdm\_msg\_set\_boolean\_array\_name (lbmsdm\_msg\_t \* *Message*, const char \* *Name*)

**Parameters:**

*Message* The SDM message containing the field.

*Name* Field name.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

#### 6.16.2.3 LBMSDMDLL int lbmsdm\_msg\_set\_decimal\_array\_name (lbmsdm\_msg\_t \* *Message*, const char \* *Name*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

#### 6.16.2.4 LBMSDMDLL int lbmsdm\_msg\_set\_double\_array\_name (lbmsdm\_msg\_t \* *Message*, const char \* *Name*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

#### 6.16.2.5 LBMSDMDLL int lbmsdm\_msg\_set\_float\_array\_name (lbmsdm\_msg\_t \* *Message*, const char \* *Name*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.16.2.6 LBMSDMExpDLL int lbmsdm\_msg\_set\_int16\_array\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.16.2.7 LBMSDMExpDLL int lbmsdm\_msg\_set\_int32\_array\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.16.2.8 LBMSDMExpDLL int lbmsdm\_msg\_set\_int64\_array\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.16.2.9 LBMSDMExpDLL int lbmsdm\_msg\_set\_int8\_array\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.16.2.10 LBMSDMExpDLL int lbmsdm\_msg\_set\_message\_array\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.16.2.11 LBMSDMExpDLL int lbmsdm\_msg\_set\_string\_array\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.16.2.12 LBMSDMExpDLL int lbmsdm\_msg\_set\_timestamp\_array\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.16.2.13 LBMSDMDExpDLL int lbmsdm\_msg\_set\_uint16\_array\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.16.2.14 LBMSDMDExpDLL int lbmsdm\_msg\_set\_uint32\_array\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.16.2.15 LBMSDMDExpDLL int lbmsdm\_msg\_set\_uint64\_array\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.16.2.16 LBMSDMDExpDLL int lbmsdm\_msg\_set\_uint8\_array\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.16.2.17 LBMSDMDExpDLL int lbmsdm\_msg\_set\_unicode\_array\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

## **6.17 Set a field value in a message, referenced by an iterator, to an array field.**

### **Functions**

- LBMSDMExpDLL int [`lbmsdm\_iter\_set\_boolean\_array`](#) ([`lbmsdm\_iter\_t \*Iterator`](#))  
*Set a field in a message by field name to an array field.*
- LBMSDMExpDLL int [`lbmsdm\_iter\_set\_int8\_array`](#) ([`lbmsdm\_iter\_t \*Iterator`](#))
- LBMSDMExpDLL int [`lbmsdm\_iter\_set\_uint8\_array`](#) ([`lbmsdm\_iter\_t \*Iterator`](#))
- LBMSDMExpDLL int [`lbmsdm\_iter\_set\_int16\_array`](#) ([`lbmsdm\_iter\_t \*Iterator`](#))
- LBMSDMExpDLL int [`lbmsdm\_iter\_set\_uint16\_array`](#) ([`lbmsdm\_iter\_t \*Iterator`](#))
- LBMSDMExpDLL int [`lbmsdm\_iter\_set\_int32\_array`](#) ([`lbmsdm\_iter\_t \*Iterator`](#))
- LBMSDMExpDLL int [`lbmsdm\_iter\_set\_uint32\_array`](#) ([`lbmsdm\_iter\_t \*Iterator`](#))
- LBMSDMExpDLL int [`lbmsdm\_iter\_set\_int64\_array`](#) ([`lbmsdm\_iter\_t \*Iterator`](#))
- LBMSDMExpDLL int [`lbmsdm\_iter\_set\_uint64\_array`](#) ([`lbmsdm\_iter\_t \*Iterator`](#))
- LBMSDMExpDLL int [`lbmsdm\_iter\_set\_float\_array`](#) ([`lbmsdm\_iter\_t \*Iterator`](#))
- LBMSDMExpDLL int [`lbmsdm\_iter\_set\_double\_array`](#) ([`lbmsdm\_iter\_t \*Iterator`](#))
- LBMSDMExpDLL int [`lbmsdm\_iter\_set\_decimal\_array`](#) ([`lbmsdm\_iter\_t \*Iterator`](#))
- LBMSDMExpDLL int [`lbmsdm\_iter\_set\_timestamp\_array`](#) ([`lbmsdm\_iter\_t \*Iterator`](#))
- LBMSDMExpDLL int [`lbmsdm\_iter\_set\_message\_array`](#) ([`lbmsdm\_iter\_t \*Iterator`](#))
- LBMSDMExpDLL int [`lbmsdm\_iter\_set\_string\_array`](#) ([`lbmsdm\_iter\_t \*Iterator`](#))
- LBMSDMExpDLL int [`lbmsdm\_iter\_set\_unicode\_array`](#) ([`lbmsdm\_iter\_t \*Iterator`](#))
- LBMSDMExpDLL int [`lbmsdm\_iter\_set\_blob\_array`](#) ([`lbmsdm\_iter\_t \*Iterator`](#))

### **6.17.1 Detailed Description**

The functions in this group allow the type of the field to be set to an array type, for a field referenced by an iterator.

### **6.17.2 Function Documentation**

#### **6.17.2.1 LBMSDMExpDLL int [`lbmsdm\_iter\_set\_blob\_array`](#) ([`lbmsdm\_iter\_t \*Iterator`](#))**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.17.2.2 LBMSDMExpDLL int lbmsdm\_iter\_set\_boolean\_array  
(lbmsdm\_iter\_t \* *Iterator*)**

**Parameters:**

*Iterator* The iterator referencing the field.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

**6.17.2.3 LBMSDMExpDLL int lbmsdm\_iter\_set\_decimal\_array  
(lbmsdm\_iter\_t \* *Iterator*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.17.2.4 LBMSDMExpDLL int lbmsdm\_iter\_set\_double\_array (lbmsdm\_iter\_t  
\* *Iterator*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.17.2.5 LBMSDMExpDLL int lbmsdm\_iter\_set\_float\_array (lbmsdm\_iter\_t \*  
*Iterator*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.17.2.6 LBMSDMExpDLL int lbmsdm\_iter\_set\_int16\_array (lbmsdm\_iter\_t \*  
*Iterator*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.17.2.7 LBMSDMExpDLL int lbmsdm\_iter\_set\_int32\_array (lbmsdm\_iter\_t \*  
*Iterator*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

## **6.17 Set a field value in a message, referenced by an iterator, to an array field. 95**

### **6.17.2.8 LBMSDMExpDLL int lbmsdm\_iter\_set\_int64\_array (lbmsdm\_iter\_t \* *Iterator*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

### **6.17.2.9 LBMSDMExpDLL int lbmsdm\_iter\_set\_int8\_array (lbmsdm\_iter\_t \* *Iterator*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

### **6.17.2.10 LBMSDMExpDLL int lbmsdm\_iter\_set\_message\_array (lbmsdm\_iter\_t \* *Iterator*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

### **6.17.2.11 LBMSDMExpDLL int lbmsdm\_iter\_set\_string\_array (lbmsdm\_iter\_t \* *Iterator*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

### **6.17.2.12 LBMSDMExpDLL int lbmsdm\_iter\_set\_timestamp\_array (lbmsdm\_iter\_t \* *Iterator*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

### **6.17.2.13 LBMSDMExpDLL int lbmsdm\_iter\_set\_uint16\_array (lbmsdm\_iter\_t \* *Iterator*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

### **6.17.2.14 LBMSDMExpDLL int lbmsdm\_iter\_set\_uint32\_array (lbmsdm\_iter\_t \* *Iterator*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.17.2.15 LBMSDMDLL int lbmsdm\_iter\_set\_uint64\_array  
(lbmsdm\_iter\_t \* *Iterator*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.17.2.16 LBMSDMDLL int lbmsdm\_iter\_set\_uint8\_array (lbmsdm\_iter\_t  
\* *Iterator*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.17.2.17 LBMSDMDLL int lbmsdm\_iter\_set\_unicode\_array  
(lbmsdm\_iter\_t \* *Iterator*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

## 6.18 Set an array field element value by field index

### Functions

- LBMSDMExpDLL int `lbmsdm_msg_set_boolean_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, uint8\_t Value)  
*Set the value of an array field element in a message by field index.*
- LBMSDMExpDLL int `lbmsdm_msg_set_int8_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, int8\_t Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_uint8_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, uint8\_t Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_int16_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, int16\_t Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_uint16_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, uint16\_t Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_int32_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, int32\_t Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_uint32_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, uint32\_t Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_int64_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, int64\_t Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_uint64_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, uint64\_t Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_float_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, float Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_double_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, double Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_decimal_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, const `lbmsdm_decimal_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_timestamp_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, const struct timeval \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_message_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, const `lbmsdm_msg_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_string_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, const char \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_unicode_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, const wchar\_t \*Value, size\_t Length)  
*Set the value of a unicode array field element in a message by field index.*
- LBMSDMExpDLL int `lbmsdm_msg_set_blob_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, const void \*Value, size\_t Length)  
*Set the value of a BLOB array field element in a message by field index.*

### 6.18.1 Detailed Description

The functions in this group allow the value of an element of an array field to be set, for a field referenced by field index.

### 6.18.2 Function Documentation

#### 6.18.2.1 LBMSDMExpDLL int lbmsdm\_msg\_set\_blob\_elem\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, const void \* *Value*, size\_t *Length*)

##### Parameters:

*Message* The SDM message containing the field.

*Index* Field index.

*Element* Array element (zero-based).

*Value* Pointer to variable containing the element value.

*Length* Length of *Value* in bytes.

##### Returns:

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

#### 6.18.2.2 LBMSDMExpDLL int lbmsdm\_msg\_set\_boolean\_elem\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, uint8\_t *Value*)

##### Parameters:

*Message* The SDM message containing the field.

*Index* Field index.

*Element* Array element (zero-based).

*Value* Pointer to variable containing the element value.

##### Returns:

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

#### 6.18.2.3 LBMSDMExpDLL int lbmsdm\_msg\_set\_decimal\_elem\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, const lbmsdm\_decimal\_t \* *Value*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.18.2.4 LBMSDMExpDLL int lbmsdm\_msg\_set\_double\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, double *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.18.2.5 LBMSDMExpDLL int lbmsdm\_msg\_set\_float\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, float *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.18.2.6 LBMSDMExpDLL int lbmsdm\_msg\_set\_int16\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, int16\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.18.2.7 LBMSDMExpDLL int lbmsdm\_msg\_set\_int32\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, int32\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.18.2.8 LBMSDMExpDLL int lbmsdm\_msg\_set\_int64\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, int64\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.18.2.9 LBMSDMExpDLL int lbmsdm\_msg\_set\_int8\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, int8\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.18.2.10 LBMSDMExpDLL int lbmsdm\_msg\_set\_message\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, const  
lbmsdm\_msg\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.18.2.11 LBMSDMDLL int lbmsdm\_msg\_set\_string\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, const char \*  
*Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.18.2.12 LBMSDMDLL int lbmsdm\_msg\_set\_timestamp\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, const struct  
timeval \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.18.2.13 LBMSDMDLL int lbmsdm\_msg\_set\_uint16\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, uint16\_t  
*Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.18.2.14 LBMSDMDLL int lbmsdm\_msg\_set\_uint32\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, uint32\_t  
*Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.18.2.15 LBMSDMDLL int lbmsdm\_msg\_set\_uint64\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, uint64\_t  
*Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.18.2.16 LBMSDMDLL int lbmsdm\_msg\_set\_uint8\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, uint8\_t  
*Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.18.2.17 LBMSDMExpDLL int lbmsdm\_msg\_set\_unicode\_elem\_idx  
(lbmsdm\_msg\_t \* *Message*, size\_t *Index*, size\_t *Element*, const  
wchar\_t \* *Value*, size\_t *Length*)**

**Parameters:**

***Message*** The SDM message containing the field.

***Index*** Field index.

***Element*** Array element (zero-based).

***Value*** Pointer to variable containing the element value.

***Length*** Length of *Value* in wchar\_ts.

**Returns:**

**LBMSDM\_SUCCESS** if successful, **LBMSDM\_FAILURE** otherwise.

## 6.19 Set an array field element value by field name

### Functions

- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_boolean\\_elem\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, size\_t Element, uint8\_t Value)

*Set the value of an array field element in a message by field name.*

- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_int8\\_elem\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, size\_t Element, int8\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_uint8\\_elem\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, size\_t Element, uint8\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_int16\\_elem\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, size\_t Element, int16\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_uint16\\_elem\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, size\_t Element, uint16\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_int32\\_elem\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, size\_t Element, int32\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_uint32\\_elem\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, size\_t Element, uint32\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_int64\\_elem\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, size\_t Element, int64\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_uint64\\_elem\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, size\_t Element, uint64\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_float\\_elem\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, size\_t Element, float Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_double\\_elem\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, size\_t Element, double Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_decimal\\_elem\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, size\_t Element, const [lbmsdm\\_decimal\\_t](#) \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_timestamp\\_elem\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, size\_t Element, const struct timeval \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_message\\_elem\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, size\_t Element, const [lbmsdm\\_msg\\_t](#) \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_string\\_elem\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, size\_t Element, const char \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_unicode\\_elem\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, size\_t Element, const wchar\_t \*Value, size\_t Length)

*Set the value of a unicode array field element in a message by field name.*

- LBMSDMExpDLL int `lbmsdm_msg_set_blob_elem_name` (`lbmsdm_msg_t *Message`, `const char *Name`, `size_t Element`, `const void *Value`, `size_t Length`)

*Set the value of a BLOB array field element in a message by field name.*

### 6.19.1 Detailed Description

The functions in this group allow the value of an element of an array field to be set, for a field referenced by field name.

### 6.19.2 Function Documentation

#### 6.19.2.1 LBMSDMExpDLL int `lbmsdm_msg_set_blob_elem_name` (`lbmsdm_msg_t *Message`, `const char *Name`, `size_t Element`, `const void *Value`, `size_t Length`)

##### Parameters:

**Message** The SDM message containing the field.

**Name** Field name.

**Element** Array element (zero-based).

**Value** New value.

**Length** Length of *Value* in bytes.

##### Returns:

`LBMSDM_SUCCESS` if successful, `LBMSDM_FAILURE` otherwise.

#### 6.19.2.2 LBMSDMExpDLL int `lbmsdm_msg_set_boolean_elem_name` (`lbmsdm_msg_t *Message`, `const char *Name`, `size_t Element`, `uint8_t Value`)

##### Parameters:

**Message** The SDM message containing the field.

**Name** Field name.

**Element** Array element (zero-based).

**Value** Pointer to variable where the value is stored.

##### Returns:

`LBMSDM_SUCCESS` if successful, `LBMSDM_FAILURE` otherwise.

**6.19.2.3 LBMSDMDExpDLL int lbmsdm\_msg\_set\_decimal\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*, const  
lbmsdm\_decimal\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.19.2.4 LBMSDMDExpDLL int lbmsdm\_msg\_set\_double\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*, double  
*Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.19.2.5 LBMSDMDExpDLL int lbmsdm\_msg\_set\_float\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*, float  
*Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.19.2.6 LBMSDMDExpDLL int lbmsdm\_msg\_set\_int16\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*, int16\_t  
*Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.19.2.7 LBMSDMDExpDLL int lbmsdm\_msg\_set\_int32\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*, int32\_t  
*Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.19.2.8 LBMSDMDExpDLL int lbmsdm\_msg\_set\_int64\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*, int64\_t  
*Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.19.2.9 LBMSDMExpDLL int lbmsdm\_msg\_set\_int8\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*, int8\_t  
*Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.19.2.10 LBMSDMExpDLL int lbmsdm\_msg\_set\_message\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*, const  
lbmsdm\_msg\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.19.2.11 LBMSDMExpDLL int lbmsdm\_msg\_set\_string\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*, const  
char \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.19.2.12 LBMSDMExpDLL int lbmsdm\_msg\_set\_timestamp\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*, const  
struct timeval \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.19.2.13 LBMSDMExpDLL int lbmsdm\_msg\_set\_uint16\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*,  
uint16\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.19.2.14 LBMSDMExpDLL int lbmsdm\_msg\_set\_uint32\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*,  
uint32\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.19.2.15 LBMSDMDExpDLL int lbmsdm\_msg\_set\_uint64\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*,  
uint64\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.19.2.16 LBMSDMDExpDLL int lbmsdm\_msg\_set\_uint8\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*,  
uint8\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.19.2.17 LBMSDMDExpDLL int lbmsdm\_msg\_set\_unicode\_elem\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*, const  
wchar\_t \* *Value*, size\_t *Length*)**

**Parameters:**

*Message* The SDM message containing the field.

*Name* Field name.

*Element* Array element (zero-based).

*Value* New value.

*Length* Length of *Value* in wchar\_ts.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

## 6.20 Set an array field element value for a field referenced by an iterator

### Functions

- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_boolean\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, uint8\_t Value)

*Set the value of an array field element in the field referenced by an iterator.*

- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_int8\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, int8\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_uint8\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, uint8\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_int16\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, int16\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_uint16\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, uint16\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_int32\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, int32\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_uint32\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, uint32\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_int64\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, int64\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_uint64\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, uint64\_t Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_float\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, float Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_double\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, double Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_decimal\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, const [lbmsdm\\_decimal\\_t](#) \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_timestamp\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, const struct timeval \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_message\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, const [lbmsdm\\_msg\\_t](#) \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_string\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, const char \*Value)
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_unicode\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, const wchar\_t \*Value, size\_t Length)

*Set the value of a unicode array field element in the field referenced by an iterator.*

- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_blob\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, const void \*Value, size\_t Length)

*Set the value of a BLOB array field element in the field referenced by an iterator.*

### 6.20.1 Detailed Description

The functions in this group allow the value of an element of an array field to be set, for a field referenced by an iterator.

### 6.20.2 Function Documentation

#### 6.20.2.1 LBMSDMExpDLL int lbmsdm\_iter\_set\_blob\_elem (lbmsdm\_iter\_t \* *Iterator*, size\_t *Element*, const void \* *Value*, size\_t *Length*)

**Parameters:**

*Iterator* The SDM iterator to use.

*Element* Array element (zero-based).

*Value* New field value.

*Length* Length of *Value* in bytes.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

#### 6.20.2.2 LBMSDMExpDLL int lbmsdm\_iter\_set\_boolean\_elem (lbmsdm\_iter\_t \* *Iterator*, size\_t *Element*, uint8\_t *Value*)

**Parameters:**

*Iterator* The SDM iterator to use.

*Element* Array element (zero-based).

*Value* New field value.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

#### 6.20.2.3 LBMSDMExpDLL int lbmsdm\_iter\_set\_decimal\_elem (lbmsdm\_iter\_t \* *Iterator*, size\_t *Element*, const lbmsdm\_decimal\_t \* *Value*)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.20.2.4 LBMSDMExpDLL int lbmsdm\_iter\_set\_double\_elem (lbmsdm\_iter\_t \* *Iterator*, size\_t *Element*, double *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.20.2.5 LBMSDMExpDLL int lbmsdm\_iter\_set\_float\_elem (lbmsdm\_iter\_t \* *Iterator*, size\_t *Element*, float *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.20.2.6 LBMSDMExpDLL int lbmsdm\_iter\_set\_int16\_elem (lbmsdm\_iter\_t \* *Iterator*, size\_t *Element*, int16\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.20.2.7 LBMSDMExpDLL int lbmsdm\_iter\_set\_int32\_elem (lbmsdm\_iter\_t \* *Iterator*, size\_t *Element*, int32\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.20.2.8 LBMSDMExpDLL int lbmsdm\_iter\_set\_int64\_elem (lbmsdm\_iter\_t \* *Iterator*, size\_t *Element*, int64\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.20.2.9 LBMSDMExpDLL int lbmsdm\_iter\_set\_int8\_elem (lbmsdm\_iter\_t \* *Iterator*, size\_t *Element*, int8\_t *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.20.2.10 LBMSDMExpDLL int lbmsdm\_iter\_set\_message\_elem  
(lbmsdm\_iter\_t \* *Iterator*, size\_t *Element*, const lbmsdm\_msg\_t \* *Value*)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.20.2.11 LBMSDMDLL int lbmsdm\_iter\_set\_string\_elem (lbmsdm\_iter\_t \* Iterator, size\_t Element, const char \* Value)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.20.2.12 LBMSDMDLL int lbmsdm\_iter\_set\_timestamp\_elem (lbmsdm\_iter\_t \* Iterator, size\_t Element, const struct timeval \* Value)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.20.2.13 LBMSDMDLL int lbmsdm\_iter\_set\_uint16\_elem (lbmsdm\_iter\_t \* Iterator, size\_t Element, uint16\_t Value)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.20.2.14 LBMSDMDLL int lbmsdm\_iter\_set\_uint32\_elem (lbmsdm\_iter\_t \* Iterator, size\_t Element, uint32\_t Value)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.20.2.15 LBMSDMDLL int lbmsdm\_iter\_set\_uint64\_elem (lbmsdm\_iter\_t \* Iterator, size\_t Element, uint64\_t Value)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.20.2.16 LBMSDMDLL int lbmsdm\_iter\_set\_uint8\_elem (lbmsdm\_iter\_t \* Iterator, size\_t Element, uint8\_t Value)**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**6.20.2.17 LBMSDMExpDLL int lbmsdm\_iter\_set\_unicode\_elem  
(lbmsdm\_iter\_t \* *Iterator*, size\_t *Element*, const wchar\_t \* *Value*,  
size\_t *Length*)**

**Parameters:**

*Iterator* The SDM iterator to use.

*Element* Array element (zero-based).

*Value* New field value.

*Length* Length of *Value* in wchar\_ts.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.



# Chapter 7

## LBM API Data Structure Documentation

### 7.1 `lbm_apphdr_chain_elem_t_stct` Struct Reference

Structure that represents an element in an app header chain.

```
#include <lbm.h>
```

#### Data Fields

- `lbm_uchar_t type`
- `lbm_ushort_t subtype`
- `size_t len`
- `void * data`

#### 7.1.1 Field Documentation

##### 7.1.1.1 `lbm_uchar_t lbm_apphdr_chain_elem_t_stct::type`

Code representing the type of data in this element.

##### 7.1.1.2 `lbm_ushort_t lbm_apphdr_chain_elem_t_stct::subtype`

Code representing the subtype (if any) of the data in this element.

**7.1.1.3 size\_t lbm\_apphdr\_chain\_elem\_t\_stct::len**

length of the data pointed to by `data`

**7.1.1.4 void\* lbm\_apphdr\_chain\_elem\_t\_stct::data**

Pointer to the app header data

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.2 `lbtm_async_operation_func_t` Struct Reference

Structure that holds information for asynchronous operation callbacks.

```
#include <lbtm.h>
```

### Data Fields

- `lbtm_async_operation_function_cb` `func`
- `lbtm_event_queue_t *` `evq`
- `void *` `clientd`
- `int` `flags`

#### 7.2.1 Field Documentation

##### 7.2.1.1 `lbtm_async_operation_function_cb` `lbtm_async_operation_func_t::func`

A callback function to receive status and completion of an asynchronous operation.

##### 7.2.1.2 `lbtm_event_queue_t*` `lbtm_async_operation_func_t::evq`

An event queue pointer; not yet supported. Should be set to NULL.

##### 7.2.1.3 `void* lbtm_async_operation_func_t::clientd`

A client object pointer to be passed back in to the specified callback.

##### 7.2.1.4 `int lbtm_async_operation_func_t::flags`

Flags that indicate which optional portions are included and may affect callback behavior.

The documentation for this struct was generated from the following file:

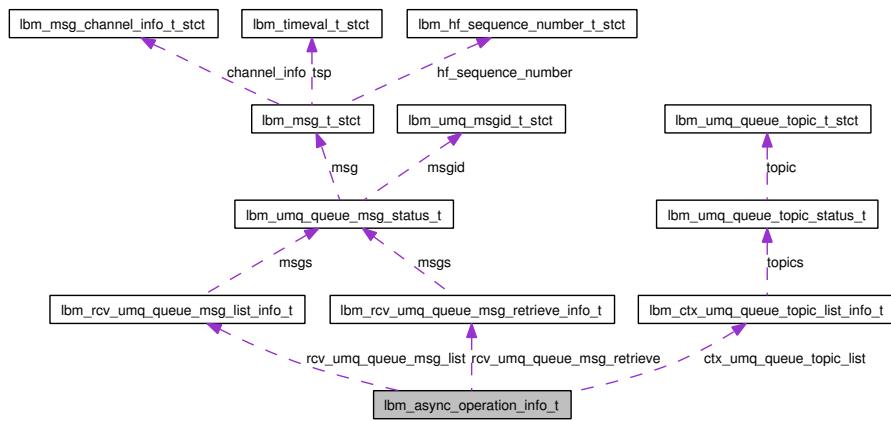
- `lbtm.h`

## 7.3 `lmb_async_operation_info_t` Struct Reference

Results struct returned via the user-specified asynchronous operation callback from any asynchronous API.

```
#include <lmb.h>
```

Collaboration diagram for `lmb_async_operation_info_t`:



### Data Fields

- int `type`
- int `status`
- int `flags`
- `lmb_async_operation_handle_t` `handle`
- union {
  - `lmb_ctx_umq_queue_topic_list_info_t` \* `ctx_umq_queue_topic_list`
  - `lmb_rcv_umq_queue_msg_list_info_t` \* `rcv_umq_queue_msg_list`
  - `lmb_rcv_umq_queue_msg_retrieve_info_t` \* `rcv_umq_queue_msg_retrieve`
} `info`

#### 7.3.1 Detailed Description

See also:

[LBM\\_ASYNC\\_OP\\_TYPE\\_CTX\\_UMQ\\_QUEUE\\_TOPIC\\_LIST](#)  
[LBM\\_ASYNC\\_OP\\_TYPE\\_RCV\\_UMQ\\_QUEUE\\_MSG\\_LIST](#)  
[LBM\\_ASYNC\\_OP\\_TYPE\\_RCV\\_UMQ\\_QUEUE\\_MSG\\_RETRIEVE](#)

### 7.3.2 Field Documentation

#### 7.3.2.1 `int lbt_async_operation_info_t::type`

The type of asynchronous operation.

#### 7.3.2.2 `int lbt_async_operation_info_t::status`

The current status of the operation.

#### 7.3.2.3 `int lbt_async_operation_info_t::flags`

Flags with extra information about the async operation.

#### 7.3.2.4 `lbt_async_operation_handle_t lbt_async_operation_info_t::handle`

An opaque handle to the asynchronous operation.

#### 7.3.2.5 `union { ... } lbt_async_operation_info_t::info`

Operation-specific results.

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.4 `lbt_context_event_func_t_stct` Struct Reference

Structure that holds the application callback for context-level events.

```
#include <lbt.h>
```

### Data Fields

- `lbt_context_event_cb_proc func`
- `lbt_event_queue_t * evq`
- `void * clientd`

#### 7.4.1 Detailed Description

A struct used to set a context-level event callback and callback info.

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.5 `lmb_context_event_umq_registration_complete_ex_t_stct` Struct Reference

### **7.5 `lmb_context_event_umq_registration_complete_ex_t_stct` Struct Reference**

Structure that holds information for contexts after registration is complete to all involved queue instances.

```
#include <lmb.h>
```

#### **Data Fields**

- int `flags`
- `lmb_umq_regid_t registration_id`
- `lmb_uint_t queue_id`
- char `queue` [LBM\_UMQ\_MAX\_QUEUE\_STRLEN]

#### **7.5.1 Detailed Description**

A structure used with UMQ receivers and sources to indicate successful context registration to quorum or to all queue instances involved.

#### **7.5.2 Field Documentation**

##### **7.5.2.1 int `lmb_context_event_umq_registration_complete_ex_t_stct::flags`**

Flags that indicate which optional portions are included

##### **7.5.2.2 `lmb_umq_regid_t lmb_context_event_umq_registration_complete_ex_t_stct::registration_id`**

Registration ID used for the registration

##### **7.5.2.3 `lmb_uint_t lmb_context_event_umq_registration_complete_ex_t_stct::queue_id`**

The Queue ID of the queue

##### **7.5.2.4 `char lmb_context_event_umq_registration_complete_ex_t_stct::queue[LBM_UMQ_MAX_QUEUE_STRLEN]`**

The name of the queue registered with

The documentation for this struct was generated from the following file:

- [lbtm.h](#)

## 7.6 `lmb_context_event_umq_registration_ex_t_stct` Struct Reference

Structure that holds queue registration information for the UMQ context in an extended form.

```
#include <lmb.h>
```

### Data Fields

- `int flags`
- `lmb_umq_regid_t registration_id`
- `lmb_uint_t queue_id`
- `lmb_uint_t queue_instance_index`
- `char queue_instance [LBM_UME_MAX_STORE_STRLEN]`
- `char queue [LBM_UMQ_MAX_QUEUE_STRLEN]`

### 7.6.1 Detailed Description

A structure used with UMQ receivers and sources to indicate successful context registration with an instance of the queue.

### 7.6.2 Field Documentation

#### 7.6.2.1 `int lmb_context_event_umq_registration_ex_t_stct::flags`

Flags that indicate which optional portions are included

#### 7.6.2.2 `lmb_umq_regid_t lmb_context_event_umq_registration_ex_t_stct::registration_id`

Registration ID used for the registration

#### 7.6.2.3 `lmb_uint_t lmb_context_event_umq_registration_ex_t_stct::queue_id`

The Queue ID of the queue

#### 7.6.2.4 `lmb_uint_t lmb_context_event_umq_registration_ex_t_stct::queue_instance_index`

The index of the instance of the queue registered with

**7.6.2.5 char lbm\_context\_event\_umq\_registration\_ex\_t\_stct::queue\_instance[LBM\_UME\_MAX\_STORE\_STRLEN]**

The instance of the queue registered with

**7.6.2.6 char lbm\_context\_event\_umq\_registration\_ex\_t\_stct::queue[LBM\_UME\_MAX\_QUEUE\_STRLEN]**

The name of the queue registered with

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.7 **lbt\_context\_rcv\_immediate\_msgs\_func\_t\_stct** Struct Reference

Structure that holds the application callback for receiving topic-less immediate mode messages.

```
#include <lbt.h>
```

### Data Fields

- [lbt\\_immediate\\_msg\\_cb\\_proc func](#)
- [lbt\\_event\\_queue\\_t \\* evq](#)
- [void \\* clientd](#)

#### 7.7.1 Detailed Description

A struct used to set the context-level topic-less immediate mode message receiver call-back. If an event queue is specified, messages will be placed on the event queue; if evq is NULL, messages will be delivered directly from the context thread.

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.8 `lbt_context_src_event_func_t_stct` Struct Reference

Structure that holds the application callback for context-level source events.

```
#include <lbt.h>
```

### Data Fields

- `lbt_context_src_cb_proc func`
- `lbt_event_queue_t * evq`
- `void * clientd`

#### 7.8.1 Detailed Description

A struct used to set a context-level source event callback and callback info.

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.9 `lbtm_context_stats_t_stct` Struct Reference

Structure that holds statistics for a context.

```
#include <lbtm.h>
```

### Data Fields

- `lbtm_ulong_t tr_dgrams_sent`
- `lbtm_ulong_t tr_bytes_sent`
- `lbtm_ulong_t tr_dgrams_rcved`
- `lbtm_ulong_t tr_bytes_rcved`
- `lbtm_ulong_t tr_dgrams_dropped_ver`
- `lbtm_ulong_t tr_dgrams_dropped_type`
- `lbtm_ulong_t tr_dgrams_dropped_malformed`
- `lbtm_ulong_t tr_dgrams_send_failed`
- `lbtm_ulong_t tr_src_topics`
- `lbtm_ulong_t tr_rcv_topics`
- `lbtm_ulong_t tr_rcv_unresolved_topics`
- `lbtm_ulong_t lbtrm_unknown_msgs_rcved`
- `lbtm_ulong_t lbtru_unknown_msgs_rcved`
- `lbtm_ulong_t send_blocked`
- `lbtm_ulong_t send_would_block`
- `lbtm_ulong_t resp_blocked`
- `lbtm_ulong_t resp_would_block`
- `lbtm_ulong_t uim_dup_msgs_rcved`
- `lbtm_ulong_t uim_msgs_no_stream_rcved`

### 7.9.1 Detailed Description

This structure holds general context statistics for things like topic resolution and interaction with transports and applications.

### 7.9.2 Field Documentation

#### 7.9.2.1 `lbtm_ulong_t lbtm_context_stats_t_stct::tr_dgrams_sent`

Number of topic resolution datagrams sent from this context. Each datagram can contain one or more advertisements, queries, query responses, etc. from source or receiver objects. A faster accumulation of counts typically indicates more source, receiver, and/or context objects are being created.

#### 7.9.2.2 **lbtm\_ulong\_t lbtm\_context\_stats\_t\_stct::tr\_bytes\_sent**

Number of topic resolution datagram bytes sent. This count is triggered under the same circumstances as tr\_dgrams\_sent (above), but measures the total number of bytes for all datagrams sent, including their headers.

#### 7.9.2.3 **lbtm\_ulong\_t lbtm\_context\_stats\_t\_stct::tr\_dgrams\_rcved**

Number of topic resolution datagrams received by this context. Each datagram can contain one or more advertisements, queries, query responses, etc. from source or receiver objects. A faster accumulation of counts typically indicates more source, receiver, and/or context objects are being created.

#### 7.9.2.4 **lbtm\_ulong\_t lbtm\_context\_stats\_t\_stct::tr\_bytes\_rcved**

Number of topic resolution datagram bytes received. This count is triggered under the same circumstances as tr\_dgrams\_rcved (above), but measures the total number of bytes for all datagrams received, including their headers.

#### 7.9.2.5 **lbtm\_ulong\_t lbtm\_context\_stats\_t\_stct::tr\_dgrams\_dropped\_ver**

Number of topic resolution datagrams discarded due to incorrect version. The datagram's version field must match the expectations of the receiving context.

#### 7.9.2.6 **lbtm\_ulong\_t lbtm\_context\_stats\_t\_stct::tr\_dgrams\_dropped\_type**

Number of topic resolution datagrams discarded due to incorrect type. The datagram's type field must match the expectations of the receiving context.

#### 7.9.2.7 **lbtm\_ulong\_t lbtm\_context\_stats\_t\_stct::tr\_dgrams\_dropped\_malformed**

Number of topic resolution datagrams discarded due to being malformed or corrupted.

#### 7.9.2.8 **lbtm\_ulong\_t lbtm\_context\_stats\_t\_stct::tr\_dgrams\_send\_failed**

Number of topic resolution datagram sends that failed. This count should be at or at least near 0.

**7.9.2.9    lbt\_ulong\_t lbt\_context\_stats\_t\_stct::tr\_src\_topics**

Number of topics in the source topic resolver cache (also referred to as the topic map). Inordinately large or growing values here may impact performance.

**7.9.2.10    lbt\_ulong\_t lbt\_context\_stats\_t\_stct::tr\_rcv\_topics**

Total number of topics in the receiver topic resolver cache (also referred to as the topic map). Inordinately large or growing values here may impact performance.

**7.9.2.11    lbt\_ulong\_t lbt\_context\_stats\_t\_stct::tr\_rcv\_unresolved\_topics**

Number of unresolved topics in the receiver topic resolver cache (aka topic map). Inordinately large or growing values here may impact performance, though this count can be close to the total number of topics in the resolver cache under normal conditions.

**7.9.2.12    lbt\_ulong\_t lbt\_context\_stats\_t\_stct::lbtrm\_unknown\_msgs\_rcved**

Number of LBT-RM datagrams received not belonging to any transport session. Such occurrences should be investigated. These datagrams can be from a source in a different topic resolution domain targeting the same group (or IP) and port as a source of interest on this receiver's topic resolution domain. Among less likely possibilities would be an attempt to spoof UM messages.

**7.9.2.13    lbt\_ulong\_t lbt\_context\_stats\_t\_stct::lbtru\_unknown\_msgs\_rcved**

Number of LBT-RU datagrams received not belonging to any transport session. Such occurrences should be investigated. These datagrams can be from a source in a different topic resolution domain targeting the same group (or IP) and port as a source of interest on this receiver's topic resolution domain. Among less likely possibilities would be an attempt to spoof UM messages.

**7.9.2.14    lbt\_ulong\_t lbt\_context\_stats\_t\_stct::send\_blocked**

Number of incidents where a UM send call was blocked. Unusually high counts could indicate performance degradation or I/O problems.

**7.9.2.15    lbt\_ulong\_t lbt\_context\_stats\_t\_stct::send\_would\_block**

Number of incidents where a UM send call returned EWOULDBLOCK. This is when a send call set to be nonblocking encounters an error condition where it would otherwise be blocked. Under normal operating conditions, this count should be at or near 0.

**7.9.2.16 `lbt_ulong_t lbt_context_stats_t_stct::resp_blocked`**

Number of incidents where a UM send response call was blocked. Unusually high counts could indicate performance degradation or I/O problems.

**7.9.2.17 `lbt_ulong_t lbt_context_stats_t_stct::resp_would_block`**

Number of incidents where a UM send response call returned EWOULDBLOCK. This is when a send response call set as nonblocking encounters an error condition where it would otherwise be blocked. Under normal operating conditions, this count should be at or near 0.

**7.9.2.18 `lbt_ulong_t lbt_context_stats_t_stct::uim_dup_msgs_rcved`**

Number of duplicate unicast immediate messages (UIMs) received and dropped.

**7.9.2.19 `lbt_ulong_t lbt_context_stats_t_stct::uim_msgs_no_stream_rcved`**

Number of unicast immediate messages (UIMs) received without stream information.

The documentation for this struct was generated from the following file:

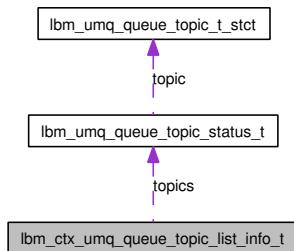
- [lbt.h](#)

## 7.10 `lbtm_ctx_umq_queue_topic_list_info_t` Struct Reference

Struct containing an array of queue topics retrieved via `lbtm_umq_queue_topic_list`.

```
#include <lbtm.h>
```

Collaboration diagram for `lbtm_ctx_umq_queue_topic_list_info_t`:



### Data Fields

- `lbtm_umq_queue_topic_status_t * topics`
- `int num_topics`

#### 7.10.1 Field Documentation

##### 7.10.1.1 `lbtm_umq_queue_topic_status_t* lbtm_ctx_umq_queue_topic_list_info_t::topics`

An array of UMQ topic status objects.

##### 7.10.1.2 `int lbtm_ctx_umq_queue_topic_list_info_t::num_topics`

The length, in number of elements, of the topic objects array.

The documentation for this struct was generated from the following file:

- [lbtm.h](#)

## 7.11 **lbtm\_delete\_cb\_info\_t\_stct** Struct Reference

Structure passed to the [lbtm\\_hypertopic\\_rcv\\_delete\(\)](#) function so that a deletion callback may be called.

```
#include <lbtmht.h>
```

### Data Fields

- [lbtm\\_delete\\_cb\\_proc](#) cbproc
- [void \\*](#) clientd

#### 7.11.1 Field Documentation

##### 7.11.1.1 [lbtm\\_delete\\_cb\\_proc](#) lbtm\_delete\_cb\_info\_t\_stct::cbproc

The cancel callback function

##### 7.11.1.2 [void\\* lbtm\\_delete\\_cb\\_info\\_t\\_stct::clientd](#)

Client Data passed in the deletion callback when called

The documentation for this struct was generated from the following file:

- [lbtmht.h](#)

## 7.12 `lbtm_event_queue_cancel_cb_info_t_stct` Struct Reference

Structure passed to cancel/delete functions so that a cancel callback may be called.

```
#include <lbtm.h>
```

### Data Fields

- `lbtm_event_queue_t * event_queue`
- `lbtm_event_queue_cancel_cb_proc cbproc`
- `void * clientd`

#### 7.12.1 Field Documentation

##### 7.12.1.1 `lbtm_event_queue_t* lbtm_event_queue_cancel_cb_info_t_stct::event_queue`

The event queue for the cancel callback

##### 7.12.1.2 `lbtm_event_queue_cancel_cb_proc lbtm_event_queue_cancel_cb_info_t_stct::cbproc`

The cancel callback function

##### 7.12.1.3 `void* lbtm_event_queue_cancel_cb_info_t_stct::clientd`

Client Data passed in the cancel callback when called

The documentation for this struct was generated from the following file:

- `lbtm.h`

## 7.13 `lmb_event_queue_stats_t_stct` Struct Reference

Structure that holds statistics for an event queue.

```
#include <lmb.h>
```

### Data Fields

- `lmb_ulong_t data_msgs`
- `lmb_ulong_t data_msgs_tot`
- `lmb_ulong_t data_msgs_svc_min`
- `lmb_ulong_t data_msgs_svc_mean`
- `lmb_ulong_t data_msgs_svc_max`
- `lmb_ulong_t resp_msgs`
- `lmb_ulong_t resp_msgs_tot`
- `lmb_ulong_t resp_msgs_svc_min`
- `lmb_ulong_t resp_msgs_svc_mean`
- `lmb_ulong_t resp_msgs_svc_max`
- `lmb_ulong_t topicless_im_msgs`
- `lmb_ulong_t topicless_im_msgs_tot`
- `lmb_ulong_t topicless_im_msgs_svc_min`
- `lmb_ulong_t topicless_im_msgs_svc_mean`
- `lmb_ulong_t topicless_im_msgs_svc_max`
- `lmb_ulong_t wrccv_msgs`
- `lmb_ulong_t wrccv_msgs_tot`
- `lmb_ulong_t wrccv_msgs_svc_min`
- `lmb_ulong_t wrccv_msgs_svc_mean`
- `lmb_ulong_t wrccv_msgs_svc_max`
- `lmb_ulong_t io_events`
- `lmb_ulong_t io_events_tot`
- `lmb_ulong_t io_events_svc_min`
- `lmb_ulong_t io_events_svc_mean`
- `lmb_ulong_t io_events_svc_max`
- `lmb_ulong_t timer_events`
- `lmb_ulong_t timer_events_tot`
- `lmb_ulong_t timer_events_svc_min`
- `lmb_ulong_t timer_events_svc_mean`
- `lmb_ulong_t timer_events_svc_max`
- `lmb_ulong_t source_events`
- `lmb_ulong_t source_events_tot`
- `lmb_ulong_t source_events_svc_min`
- `lmb_ulong_t source_events_svc_mean`
- `lmb_ulong_t source_events_svc_max`

- `lmb_ulong_t unblock_events`
- `lmb_ulong_t unblock_events_tot`
- `lmb_ulong_t cancel_events`
- `lmb_ulong_t cancel_events_tot`
- `lmb_ulong_t cancel_events_svc_min`
- `lmb_ulong_t cancel_events_svc_mean`
- `lmb_ulong_t cancel_events_svc_max`
- `lmb_ulong_t context_source_events`
- `lmb_ulong_t context_source_events_tot`
- `lmb_ulong_t context_source_events_svc_min`
- `lmb_ulong_t context_source_events_svc_mean`
- `lmb_ulong_t context_source_events_svc_max`
- `lmb_ulong_t events`
- `lmb_ulong_t events_tot`
- `lmb_ulong_t age_min`
- `lmb_ulong_t age_mean`
- `lmb_ulong_t age_max`
- `lmb_ulong_t callback_events`
- `lmb_ulong_t callback_events_tot`
- `lmb_ulong_t callback_events_svc_min`
- `lmb_ulong_t callback_events_svc_mean`
- `lmb_ulong_t callback_events_svc_max`

### 7.13.1 Detailed Description

This structure holds statistics for messages and other events that enter and exit the event queue. NOTE: Specific count-enable options must sometimes be enabled for these statistics to populate.

### 7.13.2 Field Documentation

#### 7.13.2.1 `lmb_ulong_t lmb_event_queue_stats_t_stct::data_msgs`

Number of data messages currently in the event queue, i.e., a snapshot. Configuration option `queue_count_enabled` must be activated.

#### 7.13.2.2 `lmb_ulong_t lmb_event_queue_stats_t_stct::data_msgs_tot`

Total accumulated number of data messages that have been added to the event queue (even if subsequently de-queued) since last reset. Configuration option `queue_count_enabled` must be activated.

#### 7.13.2.3 **lbt\_ulong\_t lbt\_event\_queue\_stats\_t\_stct::data\_msgs\_svc\_min**

Minimum service time for data messages (in microseconds). This is the low-water mark (i.e., the shortest so far) for data message service durations, measured from the point of de-queueument until the application has finished servicing the message. Configuration option queue\_service\_time\_enabled must be activated.

#### 7.13.2.4 **lbt\_ulong\_t lbt\_event\_queue\_stats\_t\_stct::data\_msgs\_svc\_mean**

Mean service time for data messages (in microseconds). This is an exponentially weighted moving average (weighted to more recent) for accumulated data message service durations, measured from the point of de-queueument until the application has finished servicing the message. Configuration option queue\_service\_time\_enabled must be activated.

#### 7.13.2.5 **lbt\_ulong\_t lbt\_event\_queue\_stats\_t\_stct::data\_msgs\_svc\_max**

Maximum service time for data messages (in microseconds). This is the high-water mark (i.e., the longest so far) for data message service durations measured from the point of de-queueument until the application has finished servicing the message. Configuration option queue\_service\_time\_enabled must be activated.

#### 7.13.2.6 **lbt\_ulong\_t lbt\_event\_queue\_stats\_t\_stct::resp\_msgs**

Number of response messages (from receiver objects) currently in the event queue, i.e., a snapshot. Configuration option queue\_count\_enabled must be activated.

#### 7.13.2.7 **lbt\_ulong\_t lbt\_event\_queue\_stats\_t\_stct::resp\_msgs\_tot**

Total accumulated number of response messages that have been added to the event queue (even if subsequently de-queued) since last reset.

#### 7.13.2.8 **lbt\_ulong\_t lbt\_event\_queue\_stats\_t\_stct::resp\_msgs\_svc\_min**

Minimum service time for response messages (in microseconds). This is the low-water mark (i.e., the shortest so far) for response message service durations, measured from the point of de-queueument until the application has finished servicing the message. Configuration option queue\_service\_time\_enabled must be activated.

**7.13.2.9 `lbtm_ulong_t lbtm_event_queue_stats_t_stct::resp_msgs_svc_mean`**

Mean service time for response messages (in microseconds). This is an exponentially weighted moving average (weighted to more recent) for accumulated response message service durations, measured from the point of de-queueuemnt until the application has finished servicing the message. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.10 `lbtm_ulong_t lbtm_event_queue_stats_t_stct::resp_msgs_svc_max`**

Maximum service time for response messages (in microseconds). This is the high-water mark (i.e., the longest so far) for response message service durations measured from the point of de-queueuemnt until the application has finished servicing the message. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.11 `lbtm_ulong_t lbtm_event_queue_stats_t_stct::topicless_im_msgs`**

Number of topic-less Multicast Immediate Messaging (MIM) messages currently in the event queue, i.e., a snapshot. Configuration option `queue_count_enabled` must be activated.

**7.13.2.12 `lbtm_ulong_t lbtm_event_queue_stats_t_stct::topicless_im_msgs_tot`**

Total accumulated number of topic-less Multicast Immediate Messaging (MIM) messages that have been added to the event queue (even if subsequently de-queued) since last reset. Configuration option `queue_count_enabled` must be activated.

**7.13.2.13 `lbtm_ulong_t lbtm_event_queue_stats_t_stct::topicless_im_msgs_svc_min`**

Minimum service time for topic-less Multicast Immediate Messaging (MIM) messages (in microseconds). This is the low-water mark (i.e., the shortest so far) for topic-less MIM message service durations, measured from the point of de-queueuemnt until the application has finished servicing the message. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.14 `lbtm_ulong_t lbtm_event_queue_stats_t_stct::topicless_im_msgs_svc_mean`**

Mean service time for topic-less Multicast Immediate Messaging (MIM) messages (in microseconds). This is an exponentially weighted moving average (weighted to more recent) for accumulated topic-less MIM message service durations, measured from

the point of de-queuement until the application has finished servicing the message. Configuration option queue\_service\_time\_enabled must be activated.

#### 7.13.2.15 `lbt_ulong_t lbt_event_queue_stats_t_stct::topicless_im_msgs_svc_max`

Maximum service time for topic-less Multicast Immediate Messaging (MIM) messages (in microseconds). This is the high-water mark (i.e., the longest so far) for topic-less MIM message service durations measured from the point of de-queuement until the application has finished servicing the message. Configuration option queue\_service\_time\_enabled must be activated.

#### 7.13.2.16 `lbt_ulong_t lbt_event_queue_stats_t_stct::wrcv_msgs`

Number of wildcard receiver messages currently in the event queue, i.e., a snapshot. Configuration option queue\_count\_enabled must be activated.

#### 7.13.2.17 `lbt_ulong_t lbt_event_queue_stats_t_stct::wrcv_msgs_tot`

Total accumulated number of wildcard receiver messages that have been added to the event queue (even if subsequently de-queued) since last reset. Configuration option queue\_count\_enabled must be activated.

#### 7.13.2.18 `lbt_ulong_t lbt_event_queue_stats_t_stct::wrcv_msgs_svc_min`

Minimum service time for wildcard receiver messages (in microseconds). This is the low-water mark (i.e., the shortest so far) for wildcard receiver message service durations measured from the point of de-queuement until the application has finished servicing the message. Configuration option queue\_service\_time\_enabled must be activated.

#### 7.13.2.19 `lbt_ulong_t lbt_event_queue_stats_t_stct::wrcv_msgs_svc_mean`

Mean service time for wildcard receiver messages (in microseconds). This is an exponentially weighted moving average (weighted to more recent) for accumulated wildcard receiver message service durations, measured from the point of de-queuement until the application has finished servicing the message. Configuration option queue\_service\_time\_enabled must be activated.

**7.13.2.20 `lmb_ulong_t lmb_event_queue_stats_t_stct::wrcv_msgs_svc_max`**

Maximum service time for wildcard receiver messages (in microseconds). This is the high-water mark (i.e., the longest so far) for wildcard receiver message service durations measured from the point of de-queueumet until the application has finished servicing the message. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.21 `lmb_ulong_t lmb_event_queue_stats_t_stct::io_events`**

Number of I/O events currently in the event queue, i.e., a snapshot. Configuration option `queue_count_enabled` must be activated.

**7.13.2.22 `lmb_ulong_t lmb_event_queue_stats_t_stct::io_events_tot`**

Total accumulated number of I/O events that have been added to the event queue (even if subsequently de-queued) since last reset. Configuration option `queue_count_enabled` must be activated.

**7.13.2.23 `lmb_ulong_t lmb_event_queue_stats_t_stct::io_events_svc_min`**

Minimum service time for I/O events (in microseconds). This is the low-water mark (i.e., the shortest so far) for I/O event service durations measured from the point of de-queueumet until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.24 `lmb_ulong_t lmb_event_queue_stats_t_stct::io_events_svc_mean`**

Mean service time for I/O events (in microseconds). This is an exponentially weighted moving average (weighted to more recent) for accumulated I/O event service durations, measured from the point of de-queueumet until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.25 `lmb_ulong_t lmb_event_queue_stats_t_stct::io_events_svc_max`**

Maximum service time for I/O events (in microseconds). This is the high-water mark (i.e., the longest so far) for I/O event service durations measured from the point of de-queueumet until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.26 `lbt_ulong_t lbt_event_queue_stats_t_stct::timer_events`**

Number of timer events currently in the event queue, i.e., a snapshot. Configuration option `queue_count_enabled` must be activated.

**7.13.2.27 `lbt_ulong_t lbt_event_queue_stats_t_stct::timer_events_tot`**

Total accumulated number of timer events that have been added to the event queue (even if subsequently de-queued) since last reset. Configuration option `queue_count_enabled` must be activated.

**7.13.2.28 `lbt_ulong_t lbt_event_queue_stats_t_stct::timer_events_svc_min`**

Minimum service time for timer events (in microseconds). This is the low-water mark (i.e., the shortest so far) for timer event service durations measured from the point of de-queuemt until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.29 `lbt_ulong_t lbt_event_queue_stats_t_stct::timer_events_svc_mean`**

Mean service time for timer events (in microseconds). This is an exponentially weighted moving average (weighted to more recent) for accumulated timer event service durations, measured from the point of de-queuemt until the application has finished servicing the message. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.30 `lbt_ulong_t lbt_event_queue_stats_t_stct::timer_events_svc_max`**

Maximum service time for timer events (in microseconds). This is the high-water mark (i.e., the longest so far) for timer event service durations measured from the point of de-queuemt until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.31 `lbt_ulong_t lbt_event_queue_stats_t_stct::source_events`**

Number of source events currently in the event queue, i.e., a snapshot. Configuration option `queue_count_enabled` must be activated.

**7.13.2.32 `lbt_ulong_t lbt_event_queue_stats_t_stct::source_events_tot`**

Total accumulated number of source events that have been added to the event queue (even if subsequently de-queued) since last reset. Configuration option `queue_count_`

enabled must be activated.

#### 7.13.2.33 `lmb_ulong_t lmb_event_queue_stats_t_stct::source_events_svc_min`

Minimum service time for source events (in microseconds). This is the low-water mark (i.e., the shortest so far) for source event service durations measured from the point of de-queueumet until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

#### 7.13.2.34 `lmb_ulong_t lmb_event_queue_stats_t_stct::source_events_svc_mean`

Mean service time for source events (in microseconds). This is an exponentially weighted moving average (weighted to more recent) for accumulated source event service durations, measured from the point of de-queueumet until the application has finished servicing the message. Configuration option `queue_service_time_enabled` must be activated.

#### 7.13.2.35 `lmb_ulong_t lmb_event_queue_stats_t_stct::source_events_svc_max`

Maximum service time for source events (in microseconds). This is the high-water mark (i.e., the longest so far) for source event service durations measured from the point of de-queueumet until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

#### 7.13.2.36 `lmb_ulong_t lmb_event_queue_stats_t_stct::unblock_events`

Number of unblock events currently in the event queue, i.e., a snapshot. Configuration option `queue_count_enabled` must be activated.

#### 7.13.2.37 `lmb_ulong_t lmb_event_queue_stats_t_stct::unblock_events_tot`

Total accumulated number of unblock events that have been added to the event queue (even if subsequently de-queued) since last reset. Configuration option `queue_count_enabled` must be activated.

#### 7.13.2.38 `lmb_ulong_t lmb_event_queue_stats_t_stct::cancel_events`

Number of cancel events currently in the event queue, i.e., a snapshot. Configuration option `queue_count_enabled` must be activated.

**7.13.2.39 `lbt_ulong_t lbt_event_queue_stats_t_stct::cancel_events_tot`**

Total accumulated number of cancel events that have been added to the event queue (even if subsequently de-queued) since last reset. Configuration option `queue_count_enabled` must be activated.

**7.13.2.40 `lbt_ulong_t lbt_event_queue_stats_t_stct::cancel_events_svc_min`**

Minimum service time for cancel events. Cancel events as seen by the event queue do not actually consume service time, so we do not recommend the general use of this counter.

**7.13.2.41 `lbt_ulong_t lbt_event_queue_stats_t_stct::cancel_events_svc_mean`**

Mean service time for cancel events. Cancel events as seen by the event queue do not actually consume service time, so we do not recommend the general use of this counter.

**7.13.2.42 `lbt_ulong_t lbt_event_queue_stats_t_stct::cancel_events_svc_max`**

Maximum service time for cancel events. Cancel events as seen by the event queue do not actually consume service time, so we do not recommend the general use of this counter.

**7.13.2.43 `lbt_ulong_t lbt_event_queue_stats_t_stct::context_source_events`**

Number of context source events currently in the event queue, i.e., a snapshot. Configuration option `queue_count_enabled` must be activated.

**7.13.2.44 `lbt_ulong_t lbt_event_queue_stats_t_stct::context_source_events_tot`**

Total accumulated number of context source events that have been added to the event queue (even if subsequently de-queued) since last reset. Configuration option `queue_count_enabled` must be activated.

**7.13.2.45 `lbt_ulong_t lbt_event_queue_stats_t_stct::context_source_events_svc_min`**

Minimum service time for context source events (in microseconds). This is the low-water mark (i.e., the shortest so far) for context source event service durations measured

from the point of de-queueuem until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

#### 7.13.2.46 `lmb_ulong_t lmb_event_queue_stats_t_stct::context_source_events_svc_mean`

Mean service time for context source events (in microseconds). This is an exponentially weighted moving average (weighted to more recent) for accumulated context source event service durations, measured from the point of de-queueuem until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

#### 7.13.2.47 `lmb_ulong_t lmb_event_queue_stats_t_stct::context_source_events_svc_max`

Maximum service time for context source events (in microseconds). This is the high-water mark (i.e., the longest so far) for context source event service durations measured from the point of de-queueuem until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

#### 7.13.2.48 `lmb_ulong_t lmb_event_queue_stats_t_stct::events`

Total number of events (including messages) currently in the event queue, i.e., a snapshot. Configuration option `queue_count_enabled` must be activated.

#### 7.13.2.49 `lmb_ulong_t lmb_event_queue_stats_t_stct::events_tot`

Total accumulated number of events (including messages) that have been added to the event queue (even if subsequently de-queued) since last reset. Configuration option `queue_count_enabled` must be activated.

#### 7.13.2.50 `lmb_ulong_t lmb_event_queue_stats_t_stct::age_min`

Minimum age of event queue entry when dequeued (in microseconds). This is the low-water mark for the measured age of any event or message (i.e., the shortest one so far) from the point of enqueueuem until de-queueuem. Configuration option `queue_age_enabled` must be activated.

#### 7.13.2.51 `lmb_ulong_t lmb_event_queue_stats_t_stct::age_mean`

Mean age of event queue entries when dequeued (in microseconds). This is an exponentially weighted moving average (weighted to more recent) for accumulated event or

message ages (measured from the point enqueueement until de-queueement). Configuration option queue\_age\_enabled must be activated.

#### **7.13.2.52 `lbt_ulong_t lbt_event_queue_stats_t_stct::age_max`**

Maximum age of event queue entry when dequeued (in microseconds). This is the high-water mark for the measured age of any event or message (i.e., the oldest one so far) from the point of enqueueement until de-queueement. Configuration option queue\_age\_enabled must be activated.

#### **7.13.2.53 `lbt_ulong_t lbt_event_queue_stats_t_stct::callback_events`**

Number of callback events currently in the event queue, i.e., a snapshot. Configuration option queue\_count\_enabled must be activated.

#### **7.13.2.54 `lbt_ulong_t lbt_event_queue_stats_t_stct::callback_events_tot`**

Total accumulated number of callback events that have been added to the event queue even if subsequently de-queued) since last reset. Configuration option queue\_count\_enabled must be activated.

#### **7.13.2.55 `lbt_ulong_t lbt_event_queue_stats_t_stct::callback_events_svc_min`**

Minimum service time for callback events (in microseconds). This is the low-water mark (i.e., the shortest so far) for callback event service durations measured from the point of de-queueement until the application has finished servicing the event. Configuration option queue\_service\_time\_enabled must be activated.

#### **7.13.2.56 `lbt_ulong_t lbt_event_queue_stats_t_stct::callback_events_svc_mean`**

Mean service time for callback events (in microseconds). This is an exponentially weighted moving average (weighted to more recent) for accumulated callback event service durations, measured from the point of de-queueement until the application has finished servicing the message. Configuration option queue\_service\_time\_enabled must be activated.

#### **7.13.2.57 `lbt_ulong_t lbt_event_queue_stats_t_stct::callback_events_svc_max`**

Maximum service time for callback events (in microseconds). This is the high-water mark (i.e., the longest so far) for callback event service durations measured from the

point of de-queuement until the application has finished servicing the event. Configuration option queue\_service\_time\_enabled must be activated.

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.14 `lbt_flight_size_inflight_t_stct` Struct Reference

Structure that holds information for source total inflight messages and bytes.

```
#include <lbt.h>
```

### Data Fields

- int `messages`
- lbt\_uint64\_t `bytes`

#### 7.14.1 Field Documentation

##### 7.14.1.1 int `lbt_flight_size_inflight_t_stct::messages`

Current amount of inflight messages

##### 7.14.1.2 lbt\_uint64\_t `lbt_flight_size_inflight_t_stct::bytes`

Current amount of inflight payload bytes

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.15 `lbt_hf_sequence_number_t_stct` Union Reference

Structure to hold a hot failover sequence number.

```
#include <lbt.h>
```

### Data Fields

- `lbt_uint32_t u32`
- `lbt_uint64_t u64`

#### 7.15.1 Field Documentation

##### 7.15.1.1 `lbt_uint32_t lbt_hf_sequence_number_t_stct::u32`

32 bit hot failover sequence number

##### 7.15.1.2 `lbt_uint64_t lbt_hf_sequence_number_t_stct::u64`

64 bit hot failover sequence number

The documentation for this union was generated from the following file:

- [lbt.h](#)

## 7.16 **lbm\_iovec\_t\_stct** Struct Reference

Structure, struct iovec compatible, that holds information about buffers used for vectored sends.

```
#include <lbm.h>
```

### Data Fields

- `char * iov_base`
- `size_t iov_len`

#### 7.16.1 Detailed Description

UM replacement for struct iovec for portability.

#### 7.16.2 Field Documentation

##### 7.16.2.1 `char* lbm_iovec_t_stct::iov_base`

Pointer to a segment of application data

##### 7.16.2.2 `size_t lbm_iovec_t_stct::iov_len`

Number of bytes in this segment

The documentation for this struct was generated from the following file:

- `lbm.h`

## 7.17 `lbtm_ipv4_address_mask_t_stct` Struct Reference

Structure that holds an IPv4 address and a CIDR style netmask.

```
#include <lbtm.h>
```

### Data Fields

- `lbtm_uint_t addr`
- `int bits`

#### 7.17.1 Detailed Description

A structure used with options to set/get specific addresses within a range.

#### 7.17.2 Field Documentation

##### 7.17.2.1 `lbtm_uint_t lbtm_ipv4_address_mask_t_stct::addr`

IPv4 address

##### 7.17.2.2 `int lbtm_ipv4_address_mask_t_stct::bits`

Number of leading 1's in the netmask

The documentation for this struct was generated from the following file:

- `lbtm.h`

## 7.18 **lbt\_mim\_unrecloss\_func\_t\_stct** Struct Reference

Structure that holds the application callback for multicast immediate message unrecoverable loss notification.

```
#include <lbt.h>
```

### Data Fields

- [lbt\\_mim\\_unrecloss\\_function\\_cb func](#)
- [void \\* clientd](#)

#### 7.18.1 Detailed Description

A structure used with options to set/get a specific callback function

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.19 lbm\_msg\_channel\_info\_t\_stct Struct Reference

Structure that represents UMS Spectrum channel information.

```
#include <lbm.h>
```

### Data Fields

- int [flags](#)
- lbm\_uint32\_t [channel\\_number](#)

#### 7.19.1 Detailed Description

This channel information assigns a channel designator to individual messages. Receivers may use this channel designator to filter messages or direct them to specific callbacks on a per-channel basis.

#### 7.19.2 Field Documentation

##### 7.19.2.1 int lbm\_msg\_channel\_info\_t\_stct::flags

Channel flags

- LBM\_MSG\_FLAG\_NUMBERED\_CHANNEL Message was delivered on a numbered channel.

##### 7.19.2.2 lbm\_uint32\_t lbm\_msg\_channel\_info\_t\_stct::channel\_number

Channel number in the range 0-4294967295

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.20 `lbt_msg_fragment_info_t_stct` Struct Reference

Structure that holds fragment information for UM messages when appropriate.

```
#include <lbt.h>
```

### Data Fields

- `lbt_uint_t start_sequence_number`
- `lbt_uint_t offset`
- `lbt_uint_t total_message_length`

#### 7.20.1 Detailed Description

To retrieve the UM-message fragment information held in this structure, it is typically necessary to call [lbt\\_msg\\_retrieve\\_fragment\\_info\(\)](#).

#### 7.20.2 Field Documentation

##### 7.20.2.1 `lbt_uint_t lbt_msg_fragment_info_t_stct::start_sequence_number`

The sequence number of the fragment that starts the message

##### 7.20.2.2 `lbt_uint_t lbt_msg_fragment_info_t_stct::offset`

The offset (in bytes) of this fragment from the message beginning

##### 7.20.2.3 `lbt_uint_t lbt_msg_fragment_info_t_stct::total_message_length`

The total length (in bytes) of the message this fragment is for

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.21 `lbt_msg_gateway_info_t_stct` Struct Reference

Structure that holds originating information for UM messages which arrived via a gateway.

```
#include <lbt.h>
```

### Data Fields

- `lbt_uint_t sequence_number`
- `char source [LBT_MSG_MAX_SOURCE_LEN]`

#### 7.21.1 Detailed Description

**Deprecated**

#### 7.21.2 Field Documentation

##### 7.21.2.1 `lbt_uint_t lbt_msg_gateway_info_t_stct::sequence_number`

The original sequence number (relative to the original transport.)

##### 7.21.2.2 `char lbt_msg_gateway_info_t_stct::source[LBT_MSG_MAX_SOURCE_LEN]`

The original source string.

The documentation for this struct was generated from the following file:

- `lbt.h`

## 7.22 `lbt_msg_properties_iter_t_stct` Struct Reference

A struct used for iterating over properties pointed to by an `lbt_msg_properties_t`.

```
#include <lbt.h>
```

### Data Fields

- `const char * name`
- `char * data`
- `size_t size`
- `int type`

The documentation for this struct was generated from the following file:

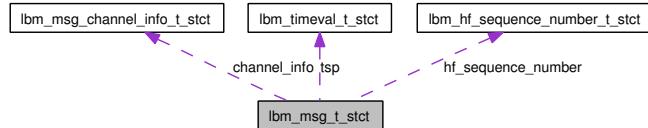
- `lbt.h`

## 7.23 lbm\_msg\_t\_stct Struct Reference

Structure that stores information about a received message.

```
#include <lbm.h>
```

Collaboration diagram for lbm\_msg\_t\_stct:



### Data Fields

- char `source` [LBM\_MSG\_MAX\_SOURCE\_LEN]
- char `topic_name` [LBM\_MSG\_MAX\_TOPIC\_LEN]
- char `copied_state` [LBM\_MSG\_MAX\_STATE\_LEN]
- int `type`
- int `flags`
- const char \* `data`
- size\_t `len`
- lbm\_response\_t \* `response`
- lbm\_uint\_t `sequence_number`
- long `refcnt`
- size\_t `apphdr_len`
- lbm\_ulong\_t `apphdr_code`
- lbm\_timeval\_t `tsp`
- lbm\_ushort\_t `hdrlen`
- const lbm\_buff\_t \* `buffer`
- const void \* `fragment_info`
- const char \* `apphdr_data`
- const void \* `source_clientd`
- const void \* `umeack`
- const void \* `src_cd`
- lbm\_uint\_t `osqn`
- lbm\_msg\_channel\_info\_t \* `channel_info`
- const void \* `umq_msqid`
- const void \* `umq_cr`
- const void \* `pdata`
- size\_t `plen`
- lbm\_msg\_properties\_t \* `properties`
- lbm\_hf\_sequence\_number\_t `hf_sequence_number`

### 7.23.1 Field Documentation

#### 7.23.1.1 `char lbm_msg_t_stct::source[LBM_MSG_MAX_SOURCE_LEN]`

Source string of transport session. Format depends on transport type. For string formats and examples, see [lbm\\_transport\\_source\\_info\\_t\\_stct](#).

#### 7.23.1.2 `char lbm_msg_t_stct::topic_name[LBM_MSG_MAX_TOPIC_LEN]`

Name of the topic. Although this field is allocated at 256 bytes, legal topic names are restricted to 246 bytes.

#### 7.23.1.3 `char lbm_msg_t_stct::copied_state[LBM_MSG_MAX_STATE_LEN]`

Copy of the state of the msg (only used on specific platforms. DO NOT ACCESS DIRECTLY!)

#### 7.23.1.4 `int lbm_msg_t_stct::type`

Type of message.

- `LBM_MSG_DATA` - Data message, Message is composed of user data
- `LBM_MSG_BOS` - Beginning of Transport Session (source connection established) (data received)
- `LBM_MSG_EOS` - End of Transport Session (connection closed to source) (no further data)
- `LBM_MSG_REQUEST` - Request message from source
- `LBM_MSG_RESPONSE` - Response message from requestee
- `LBM_MSG_UNRECOVERABLE_LOSS` - Missing message detected and not recovered in given time (no data)
- `LBM_MSG_UNRECOVERABLE_LOSS_BURST` - Missing burst of messages detected and not recovered (no data)
- `LBM_MSG_NO_SOURCE_NOTIFICATION` - No source has been found for topic, Still querying for topic source
- `LBM_MSG_UME_REGISTRATION_ERROR` - UMP receiver registration encountered an error. Data holds error message
- `LBM_MSG_UME_REGISTRATION_SUCCESS` - UMP receiver registration successful. Data holds registration IDs

- LBM\_MSG\_UME\_REGISTRATION\_CHANGE - UMP receiver notification of source registration change. Data holds info message
- LBM\_MSG\_UME\_REGISTRATION\_SUCCESS\_EX - UMP receiver registration successful for a store (extended form). Data holds registration IDs, etc.
- LBM\_MSG\_UME\_REGISTRATION\_CHANGE\_EX - UMP receiver notification of registration completion. Data holds sequence number and flags, etc.
- LBM\_MSG\_UME\_DEREGISTRATION\_SUCCESS\_EX - UMP receiver notification of deregistration success. Data holds registration IDs, etc.
- LBM\_MSG\_UME\_DEREGISTRATION\_COMPLETE\_EX - UMP receiver notification of deregistration complete.
- LBM\_MSG\_UMQ\_REGISTRATION\_ERROR - UMQ receiver registration encountered an error. Data holds error message.
- LBM\_MSG\_UMQ\_REGISTRATION\_COMPLETE\_EX - UMQ receiver notification of registration completion. Data holds Queue information, assignment ID, etc.
- LBM\_MSG\_UMQ\_DEREGISTRATION\_COMPLETE\_EX - UMQ receiver notification of de-registration completion. Data holds Queue information, etc.
- LBM\_MSG\_UMQ\_INDEX\_ASSIGNMENT\_ELIGIBILITY\_ERROR - UMQ receiver index assignment start/stop encountered an error. Data holds error message.
- LBM\_MSG\_UMQ\_INDEX\_ASSIGNMENT\_ELIGIBILITY\_START\_--COMPLETE\_EX - UMQ receiver notification of beginning of index assignment eligibility or index assignment. Data holds index information, etc.
- LBM\_MSG\_UMQ\_INDEX\_ASSIGNMENT\_ELIGIBILITY\_STOP\_--COMPLETE\_EX - UMQ receiver notification of end of index assignment eligibility or index assignment. Data holds index information, etc.
- LBM\_MSG\_UMQ\_INDEX\_ASSIGNED\_EX - UMQ receiver notification of beginning of index.
- LBM\_MSG\_UMQ\_INDEX\_RELEASED\_EX - UMQ receiver notification of end of index.
- LBM\_MSG\_UMQ\_INDEX\_ASSIGNMENT\_ERROR - UMQ receiver notification of an index assignment error.
- LBM\_MSG\_HF\_RESET - Hot-failover reset message was handled. UMS is now expecting msg->hf\_sequence\_number as the next non-reset hot-failover message.

#### 7.23.1.5 int **lbm\_msg\_t\_stct::flags**

Flags associated with the message.

- LBM\_MSG\_FLAG\_START\_BATCH - Message starts a batch
- LBM\_MSG\_FLAG\_END\_BATCH - Message ends a batch
- LBM\_MSG\_FLAG\_HF\_PASS\_THROUGH - Message is a passed-through Hot Failover message
- LBM\_MSG\_FLAG\_RETRANSMIT - Message is a late join recovered message
- LBM\_MSG\_FLAG\_UME\_RETRANSMIT - Message is a UM recovered message
- LBM\_MSG\_FLAG\_IMMEDIATE - Message is an immediate message
- LBM\_MSG\_FLAG\_TOPICLESS - Message has no topic
- LBM\_MSG\_FLAG\_HF\_32 - Message has a 32 bit hot failover sequence number
- LBM\_MSG\_FLAG\_HF\_64 - Message has a 64 bit hot failover sequence number

#### 7.23.1.6 const char\* **lbm\_msg\_t\_stct::data**

Data contents of the message if of a message type that carries data. Note that UM does not guarantee any alignment of that data.

#### 7.23.1.7 size\_t **lbm\_msg\_t\_stct::len**

Length of data in bytes

#### 7.23.1.8 lbm\_response\_t\* **lbm\_msg\_t\_stct::response**

Pointer to response object used for sending responses for `lbm_msg_t` request.

#### 7.23.1.9 lbm\_uint\_t **lbm\_msg\_t\_stct::sequence\_number**

Topic level sequence number of message. For fragmented messages, this is the sequence number of the final fragment comprising the message.

#### 7.23.1.10 const void\* **lbm\_msg\_t\_stct::source\_clientd**

Pointer set by `lbm_rcv_src_notification_create_function_cb` callback

**7.23.1.11 lbt\_msg\_channel\_info\_t\* lbt\_msg\_t\_stct::channel\_info**

Channel information set when using Spectrum channels

**7.23.1.12 lbt\_msg\_properties\_t\* lbt\_msg\_t\_stct::properties**

Message properties structure for this message

**7.23.1.13 lbt\_hf\_sequence\_number\_t lbt\_msg\_t\_stct::hf\_sequence\_number**

Hot failover sequence number, check message flags for LBM\_MSG\_FLAG\_HF\_32 or LBM\_MSG\_FLAG\_HF\_64.

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.24 `lbt_msg_ume_deregistration_ex_t_stct` Struct Reference

Structure that holds store deregistration information for the UM receiver in an extended form.

```
#include <lbt.h>
```

### Data Fields

- int `flags`
- lbt\_uint\_t `src_registration_id`
- lbt\_uint\_t `rcv_registration_id`
- lbt\_uint\_t `sequence_number`
- lbt\_ushort\_t `store_index`
- char `store` [LBM\_UME\_MAX\_STORE\_STRLEN]

### 7.24.1 Detailed Description

A structure used with UM receivers to indicate successful deregistration (extended form).

### 7.24.2 Field Documentation

#### 7.24.2.1 int `lbt_msg_ume_deregistration_ex_t_stct::flags`

Flags

#### 7.24.2.2 lbt\_uint\_t `lbt_msg_ume_deregistration_ex_t_stct::src_registration_id`

The registration ID for the source

#### 7.24.2.3 lbt\_uint\_t `lbt_msg_ume_deregistration_ex_t_stct::rcv_registration_id`

The registration ID for the receiver

**7.24.2.4   **lbt\_uint\_t lbt\_msg\_ume\_deregistration\_ex\_t\_stct::sequence\_number****

The sequence number for the receiver to end at as reported by the store

**7.24.2.5   **lbt\_ushort\_t lbt\_msg\_ume\_deregistration\_ex\_t\_stct::store\_index****

The store index of the store involved

**7.24.2.6   **char lbt\_msg\_ume\_deregistration\_ex\_t\_stct::store[LBM\_UME\_MAX\_STORE\_STRLEN]****

The store that was registered with

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.25 `lbt_msg_ume_registration_complete_ex_t_stct` Struct Reference

Structure that holds information for receivers after registration is complete to all involved stores.

```
#include <lbt.h>
```

### Data Fields

- int `flags`
- lbt\_uint\_t `sequence_number`

#### 7.25.1 Detailed Description

A structure used with UM receivers to indicate successful registration to quorum or to all stores involved.

#### 7.25.2 Field Documentation

##### 7.25.2.1 int `lbt_msg_ume_registration_complete_ex_t_stct::flags`

Flags

##### 7.25.2.2 lbt\_uint\_t `lbt_msg_ume_registration_complete_ex_t_stct::sequence_number`

The sequence number that will be the first requested

The documentation for this struct was generated from the following file:

- `lbt.h`

## 7.26 `lbt_msg_ume_registration_ex_t_stct` Struct Reference

Structure that holds store registration information for the UM receiver in an extended form.

```
#include <lbt.h>
```

### Data Fields

- int `flags`
- `lbt_uint_t src_registration_id`
- `lbt_uint_t rcv_registration_id`
- `lbt_uint_t sequence_number`
- `lbt_ushort_t store_index`
- char `store` [LBM\_UME\_MAX\_STORE\_STRLEN]

#### 7.26.1 Detailed Description

A structure used with UM receivers to indicate successful registration (extended form).

#### 7.26.2 Field Documentation

##### 7.26.2.1 int `lbt_msg_ume_registration_ex_t_stct::flags`

Flags

##### 7.26.2.2 `lbt_uint_t lbt_msg_ume_registration_ex_t_stct::src_registration_id`

The registration ID for the source

##### 7.26.2.3 `lbt_uint_t lbt_msg_ume_registration_ex_t_stct::rcv_registration_id`

The registration ID for the receiver

##### 7.26.2.4 `lbt_uint_t lbt_msg_ume_registration_ex_t_stct::sequence_number`

The sequence number for the receiver to start at as reported by the store

**7.26.2.5    `lbt_ushort_t lbt_msg_ume_registration_ex_t_stct::store_index`**

The store index of the store involved

**7.26.2.6    `char lbt_msg_ume_registration_ex_t_stct::store[LBM_UME_MAX_STORE_STRLEN]`**

The store that was registered with

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.27 `lbt_msg_ume_registration_t_stct` Struct Reference

Structure that holds store registration information for the UMP receiver.

```
#include <lbt.h>
```

### Data Fields

- `lbt_uint_t src_registration_id`
- `lbt_uint_t rcv_registration_id`

#### 7.27.1 Detailed Description

A structure used with UMP receivers to indicate successful registration.

#### 7.27.2 Field Documentation

##### 7.27.2.1 `lbt_uint_t lbt_msg_ume_registration_t_stct::src_registration_id`

The registration ID for the source

##### 7.27.2.2 `lbt_uint_t lbt_msg_ume_registration_t_stct::rcv_registration_id`

The registration ID for the receiver

The documentation for this struct was generated from the following file:

- `lbt.h`

## 7.28 `lbt_msg_umq_deregistration_complete_ex_t_stct` Struct Reference

Structure that holds information for receivers after they de-register from a queue.

```
#include <lbt.h>
```

### Data Fields

- int `flags`
- lbt\_uint\_t `queue_id`
- char `queue` [LBM\_UMQ\_MAX\_QUEUE\_STRLEN]

#### 7.28.1 Detailed Description

A struct used with UMQ receivers to indicate successful de-registration from a queue.

#### 7.28.2 Field Documentation

##### 7.28.2.1 int `lbt_msg_umq_deregistration_complete_ex_t_stct::flags`

Flags that indicate which optional portions are included

##### 7.28.2.2 lbt\_uint\_t `lbt_msg_umq_deregistration_complete_ex_t_stct::queue_id`

The Queue ID of the queue de-registering from

The documentation for this struct was generated from the following file:

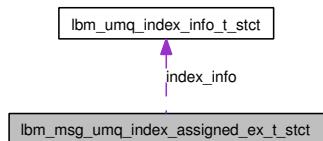
- `lbt.h`

## 7.29 `lbt_msg_umq_index_assigned_ex_t_stct` Struct Reference

Structure that holds beginning-of-index information for receivers.

```
#include <lbt.h>
```

Collaboration diagram for `lbt_msg_umq_index_assigned_ex_t_stct`:



### Data Fields

- int `flags`
- `lbt_uint_t queue_id`
- char `queue` [LBM\_UMQ\_MAX\_QUEUE\_STRLEN]
- `lbt_umq_index_info_t index_info`

#### 7.29.1 Detailed Description

A structure used with UMQ or ULB receivers to indicate the stop of index assignment from all queue instances involved.

#### 7.29.2 Field Documentation

##### 7.29.2.1 `int lbt_msg_umq_index_assigned_ex_t_stct::flags`

Flags that indicate why the index was assigned to the receiver, etc.

##### 7.29.2.2 `lbt_uint_t lbt_msg_umq_index_assigned_ex_t_stct::queue_id`

The Queue ID of the queue beginning assignment of this index

The documentation for this struct was generated from the following file:

- `lbt.h`

## 7.30 `lbt_msg_umq_index_assignment_eligibility_start_complete_ex_t_stct` Struct Reference

Structure that holds index assignment information for receivers.

```
#include <lbt.h>
```

### Data Fields

- int `flags`
- lbt\_uint\_t `queue_id`
- char `queue` [LBM\_UMQ\_MAX\_QUEUE\_STRLEN]

#### 7.30.1 Detailed Description

A structure used with UMQ or ULB receivers to indicate the start of index assignment from all queue instances involved.

#### 7.30.2 Field Documentation

##### 7.30.2.1 int `lbt_msg_umq_index_assignment_eligibility_start_complete_ex_t_stct::flags`

Flags that indicate which optional portions are included

##### 7.30.2.2 lbt\_uint\_t `lbt_msg_umq_index_assignment_eligibility_start_complete_ex_t_stct::queue_id`

The Queue ID of the queue starting index assignment eligibility

The documentation for this struct was generated from the following file:

- `lbt.h`

## 7.31 **lbt\_msg\_umq\_index\_assignment\_eligibility\_stop\_complete\_ex\_t\_stct Struct Reference**

Structure that holds index assignment information for receivers.

```
#include <lbt.h>
```

### Data Fields

- int [flags](#)
- lbt\_uint\_t [queue\\_id](#)
- char [queue](#) [LBM\_UMQ\_MAX\_QUEUE\_STRLEN]

#### 7.31.1 Detailed Description

A structure used with UMQ or ULB receivers to indicate the stop of index assignment from all queue instances involved.

#### 7.31.2 Field Documentation

##### 7.31.2.1 int [lbt\\_msg\\_umq\\_index\\_assignment\\_eligibility\\_stop\\_complete\\_ex\\_t\\_stct::flags](#)

Flags that indicate which optional portions are included

##### 7.31.2.2 lbt\_uint\_t [lbt\\_msg\\_umq\\_index\\_assignment\\_eligibility\\_stop\\_complete\\_ex\\_t\\_stct::queue\\_id](#)

The Queue ID of the queue stopping index assignment eligibility

The documentation for this struct was generated from the following file:

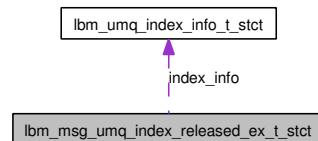
- [lbt.h](#)

## 7.32 `lbt_msg_umq_index_released_ex_t_stct` Struct Reference

Structure that holds end-of-index information for receivers.

```
#include <lbt.h>
```

Collaboration diagram for `lbt_msg_umq_index_released_ex_t_stct`:



### Data Fields

- int `flags`
- lbt\_uint\_t `queue_id`
- char `queue` [LBM\_UMQ\_MAX\_QUEUE\_STRLEN]
- `lbt_umq_index_info_t` `index_info`

#### 7.32.1 Detailed Description

A structure used with UMQ or ULB receivers to indicate the stop of index assignment from all queue instances involved.

#### 7.32.2 Field Documentation

##### 7.32.2.1 int `lbt_msg_umq_index_released_ex_t_stct::flags`

Flags that indicate why the index was released (user-requested release, etc.)

##### 7.32.2.2 lbt\_uint\_t `lbt_msg_umq_index_released_ex_t_stct::queue_id`

The Queue ID of the queue ending assignment of this index

The documentation for this struct was generated from the following file:

- `lbt.h`

## 7.33 `lbt_msg_umq_registration_complete_ex_t_stct` Struct Reference

Structure that holds information for receivers after registration is complete to all involved queue instances.

```
#include <lbt.h>
```

### Data Fields

- int `flags`
- `lbt_uint_t queue_id`
- `lbt_uint_t assignment_id`
- char `queue` [LBM\_UMQ\_MAX\_QUEUE\_STRLEN]

#### 7.33.1 Detailed Description

A structure used with UMQ receivers to indicate successful receiver registration to quorum or to all queue instances involved.

#### 7.33.2 Field Documentation

##### 7.33.2.1 int `lbt_msg_umq_registration_complete_ex_t_stct::flags`

Flags that indicate which optional portions are included

##### 7.33.2.2 `lbt_uint_t lbt_msg_umq_registration_complete_ex_t_stct::queue_id`

The Queue ID of the queue

##### 7.33.2.3 `lbt_uint_t lbt_msg_umq_registration_complete_ex_t_stct::assignment_id`

The generated Assignment ID for the receiver with the queue

##### 7.33.2.4 `char lbt_msg_umq_registration_complete_ex_t_stct::queue[LBM_UMQ_MAX_QUEUE_STRLEN]`

The name of the queue registered with

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.34 `lbt_rcv_src_notification_func_t_stct` Struct Reference

Structure that holds the application callback for source status notifications for receivers.

```
#include <lbt.h>
```

### Data Fields

- `lbt_rcv_src_notification_create_function_cb` **create\_func**
- `lbt_rcv_src_notification_delete_function_cb` **delete\_func**
- `void *` **clientd**

#### 7.34.1 Detailed Description

A structure used with options to set/get a specific callback function

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.35 **lbm\_rcv\_transport\_stats\_daemon\_t\_stct** Struct Reference

Structure that holds statistics for receiver daemon mode transport (deprecated).

```
#include <lbm.h>
```

### Data Fields

- lbm\_ulong\_t [bytes\\_rcved](#)

#### 7.35.1 Detailed Description

This structure holds statistics for receiver transports using the daemon mode. NOTE: daemon mode is deprecated and no longer available; this structure is retained for backward compatibility only.

#### 7.35.2 Field Documentation

##### 7.35.2.1 lbm\_ulong\_t **lbm\_rcv\_transport\_stats\_daemon\_t\_stct::bytes\_rcved**

This statistic has been deprecated.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.36 `lbtipc_t_stct` Struct Reference

Structure that holds datagram statistics for receiver LBT-IPC transports.

```
#include <lbtipc.h>
```

### Data Fields

- `lbtipc_t_stct::msgs_rcved`
- `lbtipc_t_stct::bytes_rcved`
- `lbtipc_t_stct::lbtipc_msgs_rcved`
- `lbtipc_t_stct::lbtipc_msgs_no_topic_rcved`
- `lbtipc_t_stct::lbtipc_reqs_rcved`

#### 7.36.1 Field Documentation

##### 7.36.1.1 `lbtipc_t_stct::msgs_rcved`

Number of LBT-IPC datagrams received. Depending on batching settings, a single LBT-IPC datagram may contain one or more messages, or a fragment of a larger message. With LBT-IPC, larger messages are split into fragment sizes limited by configuration option `transport_lbtipc_datagram_max_size` (default 64KB).

##### 7.36.1.2 `lbtipc_t_stct::bytes_rcved`

Number of LBT-IPC datagram bytes received, i.e., the total of lengths of all LBT-IPC packets including UM header information.

##### 7.36.1.3 `lbtipc_t_stct::lbtipc_msgs_rcved`

Number of messages or message fragments received over an LBT-IPC transport. A single datagram may contain one or more messages or a fragment of a larger message. For fragmented messages larger than configuration option `transport_lbtipc_datagram_max_size` (default 64KB), this count reflects the number of datagrams used to constitute those messages. Thus, this number is equal to or greater than the datagram counter (`msgs_rcved`, above). This number also includes messages received for which there was no interested receiver, which is tallied in the `lbtipc_msgs_no_topic_rcved` counter (below).

**7.36.1.4    `lbm_ulong_t lbm_recv_transport_stats_lbtipc_t_stct::lbpmsgs_no_topic_rcved`**

Number of messages received that were not for a topic of interest to the receiver. A high value (relative to, or approaching `lbpmsgs_rcved` above) indicates more CPU time required to filter out uninteresting topics, in which case, consider reconfiguring sources to filter more aggressively at the transport layer.

**7.36.1.5    `lbp_ulong_t lbm_recv_transport_stats_lbtipc_t_stct::lbpreqs_rcved`**

Number of UM request messages received (message type `LBM_MSG_REQUEST`).

The documentation for this struct was generated from the following file:

- [lbp.h](#)

## 7.37 `lbt_rcv_transport_stats_lbtrdma_t_stct` Struct Reference

Structure that holds datagram statistics for receiver LBT-RDMA transports.

```
#include <lbt.h>
```

### Data Fields

- `lbt_ulong_t msgs_rcved`
- `lbt_ulong_t bytes_rcved`
- `lbt_ulong_t lbt_msgs_rcved`
- `lbt_ulong_t lbt_msgs_no_topic_rcved`
- `lbt_ulong_t lbt_reqs_rcved`

#### 7.37.1 Field Documentation

##### 7.37.1.1 `lbt_ulong_t lbt_rcv_transport_stats_lbtrdma_t_stct::msgs_rcved`

Number of LBT-RDMA datagrams received. Depending on batching settings, a single LBT-RDMA datagram may contain one or more messages, or a fragment of a larger message. With LBT-RDMA, larger messages are split into fragment sizes limited by configuration option `transport_lbtrdma_datagram_max_size` (default 4KB).

##### 7.37.1.2 `lbt_ulong_t lbt_rcv_transport_stats_lbtrdma_t_stct::bytes_rcved`

Number of LBT-RDMA datagram bytes received, i.e., the total of lengths of all LBT-RDMA packets including UM header information.

##### 7.37.1.3 `lbt_ulong_t lbt_rcv_transport_stats_lbtrdma_t_stct::lbt_msgs_rcved`

Number of messages or message fragments received over an LBT-RDMA transport. A single datagram may contain one or more messages or a fragment of a larger message. For fragmented messages larger than configuration option `transport_lbtrdma_datagram_max_size` (default 4KB), this count reflects the number of datagrams used to constitute those messages. Thus, this number is equal to or greater than the datagram counter (`msgs_rcved`, above). This number also includes messages received for which there was no interested receiver, which is tallied in the `lbt_msgs_no_topic_rcved` counter (below).

**7.37.1.4    `lbt_ulong_t lbt_rcv_transport_stats_lbtrdma_t_stct::lbt_msgs_no_topic_rcved`**

Number of messages received that were not for a topic of interest to the receiver. A high value (relative to, or approaching `lbt_msgs_rcved` above) indicates more CPU time required to filter out uninteresting topics, in which case, consider reconfiguring sources to filter more aggressively at the transport layer.

**7.37.1.5    `lbt_ulong_t lbt_rcv_transport_stats_lbtrdma_t_stct::lbt_reqs_rcved`**

Number of UM request messages received (message type `LBM_MSG_REQUEST`).

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.38 `lbtm_rcv_transport_stats_lbtrm_t_stct` Struct Reference

Structure that holds datagram statistics for receiver LBT-RM transports.

```
#include <lbtm.h>
```

### Data Fields

- `lbtm_ulong_t msgs_rcved`
- `lbtm_ulong_t bytes_rcved`
- `lbtm_ulong_t nak_pkts_sent`
- `lbtm_ulong_t naks_sent`
- `lbtm_ulong_t lost`
- `lbtm_ulong_t ncfs_ignored`
- `lbtm_ulong_t ncfs_shed`
- `lbtm_ulong_t ncfs_rx_delay`
- `lbtm_ulong_t ncfs_unknown`
- `lbtm_ulong_t nak_stm_min`
- `lbtm_ulong_t nak_stm_mean`
- `lbtm_ulong_t nak_stm_max`
- `lbtm_ulong_t nak_tx_min`
- `lbtm_ulong_t nak_tx_mean`
- `lbtm_ulong_t nak_tx_max`
- `lbtm_ulong_t duplicate_data`
- `lbtm_ulong_t unrecovered_txw`
- `lbtm_ulong_t unrecovered_tmo`
- `lbtm_ulong_t lbtm_msgs_rcved`
- `lbtm_ulong_t lbtm_msgs_no_topic_rcved`
- `lbtm_ulong_t lbtm_reqs_rcved`
- `lbtm_ulong_t dgrams_dropped_size`
- `lbtm_ulong_t dgrams_dropped_type`
- `lbtm_ulong_t dgrams_dropped_version`
- `lbtm_ulong_t dgrams_dropped_hdr`
- `lbtm_ulong_t dgrams_dropped_other`
- `lbtm_ulong_t out_of_order`

### 7.38.1 Field Documentation

#### 7.38.1.1 `lbm_ulong_t lbm_rcv_transport_stats_lbtrm_t_stct::msgs_rcved`

Number of LBT-RM datagrams received. Depending on batching settings, a single LBT-RM datagram may contain one or more messages, or a fragment of a larger message. With LBT-RM, larger messages are split into fragment sizes limited by configuration option `transport_lbtrm_datagram_max_size` (default 8KB).

#### 7.38.1.2 `lbm_ulong_t lbm_rcv_transport_stats_lbtrm_t_stct::bytes_rcved`

Number of LBT-RM datagram bytes received, i.e., the total of lengths of all LBT-RM packets including UM header information.

#### 7.38.1.3 `lbm_ulong_t lbm_rcv_transport_stats_lbtrm_t_stct::nak_pckts_sent`

Number of NAK packets sent by the receiver transport. UM batches NAKs into NAK packets to save network bandwidth. This should always be less than or equal to the number of individual NAKs sent (`naks_sent`, below).

#### 7.38.1.4 `lbm_ulong_t lbm_rcv_transport_stats_lbtrm_t_stct::naks_sent`

Number of individual NAKs sent by the receiver transport. This may differ from the tally of lost datograms (below) due to reasons such as

- Other receiver transports may have already sent a NAK for the same lost datagram, resulting in a retransmitted lost datagram (or an NCF) to arrive at this receiver transport before it has a chance to issue a NAK, or
- During periods of heavy loss, receiver transports may be forced to issue multiple NAKs per lost datagram (controlled by configuration options `transport_lbtrm_nak_generation_interval` and `transport_lbtrm_nak_backoff_interval`) until either the retransmission is received or the datagram is declared unrecovered (which may ultimately lead to UM delivering an `LBM_MSG_UNRECOVERABLE_LOSS` notification to the receiver application).

#### 7.38.1.5 `lbm_ulong_t lbm_rcv_transport_stats_lbtrm_t_stct::lost`

Number of LBT-RM datograms detected as lost.

### 7.38.1.6 `lbtm_ulong_t lbtm_rcv_transport_stats_lbtrm_t_stct::ncfs_ignored`

Number of NCFs received from a source transport with reason code "ignored". If a source transport receives a NAK for a datagram that it has recently retransmitted, it sends an "NCF ignored" and does not retransmit. How "recently" is determined by the configuration option `source_transport_lbtrm_ignore_interval` (default 500ms). If this count is high, a receiver transport may be having trouble receiving retransmissions, or the ignore interval may be set too long.

### 7.38.1.7 `lbtm_ulong_t lbtm_rcv_transport_stats_lbtrm_t_stct::ncfs_shed`

Number of NCFs received with reason code "shed". When a source transport's retransmit queue and rate limiter are both at maximum, it responds to a NAK by sending an "NCF shed", and does not retransmit. The receiver transport should wait, then send another NAK. If this count is high, one or more crybaby receiver transports may be clogging the source transport's retransmit queue.

### 7.38.1.8 `lbtm_ulong_t lbtm_rcv_transport_stats_lbtrm_t_stct::ncfs_rx_delay`

Number of NCFs received with reason code "rx\_delay". When a source transport's retransmit rate limiter prevents it from immediately retransmitting any more lost datagrams, it responds to a NAK by sending an "NCF rx\_delay", then queues the retransmission for a later send. The receiver transport should wait for the retransmission and not immediately send another NAK. If this count is high, one or more crybaby receiver transports may be clogging the source transport's retransmit queue.

### 7.38.1.9 `lbtm_ulong_t lbtm_rcv_transport_stats_lbtrm_t_stct::ncfs_unknown`

Number of NCFs received with reason code "unknown". These are NCFs with a reason code this receiver transport does not recognize. After a delay (set by configuration option `transport_lbtrm_nak_suppress_interval` (default 1000ms), it resends the NAK. This counter should never be greater than 0 unless applications linked with different versions of Ultra Messaging software coexist on the same network.

### 7.38.1.10 `lbtm_ulong_t lbtm_rcv_transport_stats_lbtrm_t_stct::nak_stm_min`

Minimum time (in milliseconds), i.e., the shortest time recorded so far for a lost message to be recovered. If this time is greater than configuration option `transport_lbtrm_nak_backoff_interval`, it may be taking multiple NAKs to initiate retransmissions, indicating a lossy network.

**7.38.1.11 `lbtm_t_stct::nak_stm_mean`**

Mean time (in milliseconds) in which loss recovery was accomplished. This is an exponentially weighted moving average (weighted to more recent) for accumulated measured recovery times. Ideally this field should be as close to your minimum recovery time (`nak_stm_min`, above) as possible. High mean recovery times indicate a lossy network.

**7.38.1.12 `lbtm_t_stct::nak_stm_max`**

Maximum time (in milliseconds), i.e., the longest time recorded so far for a lost message to be recovered. If this time is near or equal to the configuration option `transport_lbtrm_nak_generation_interval` setting, you have likely experienced some level of unrecoverable loss.

**7.38.1.13 `lbtm_t_stct::nak_tx_min`**

Minimum number of times per lost message that a receiver transport transmitted a NAK, i.e., the lowest value collected so far. A value greater than 1 indicates a chronically lossy network.

**7.38.1.14 `lbtm_t_stct::nak_tx_mean`**

Mean number of times per lost message that a receiver transport transmitted a NAK. Ideally this should be at or near 1. A higher value indicates a lossy network. This is an exponentially weighted moving average (weighted to more recent) for accumulated NAKs per lost message.

**7.38.1.15 `lbtm_t_stct::nak_tx_max`**

Maximum number of times per lost message that a receiver transport transmitted a NAK, i.e., the highest value collected so far. A value higher than 1 suggests that there may have been some unrecoverable loss on the network during the sample period. A significantly high value (compared to the mean number) implies an isolated incident.

**7.38.1.16 `lbtm_t_stct::duplicate_data`**

Number of duplicate LBT-RM datagrams received. A large number can indicate a lossy network, primarily due to other receiver transports requesting retransmissions that this receiver transport has already successfully received. Such duplicates require extra effort for filtering, and this should be investigated.

**7.38.1.17 `lbtm_ulong_t lbtm_rev_transport_stats_lbtrm_t_stct::unrecovered_txw`**

Number of LBT-RM datagrams unrecovered (LBM\_MSG\_UNRECOVERABLE\_LOSS delivered to receiver application) due to transmission window advance. This means that the message was no longer in the source-side transmission window and therefore not retransmitted. The window size is set by transport configuration option `lbtrm_transmission_window_size` (default 24MB).

**7.38.1.18 `lbtm_ulong_t lbtm_rev_transport_stats_lbtrm_t_stct::unrecovered_tmo`**

Number of LBT-RM datagrams unrecovered due to a retransmission not received within the NAK generation interval (set by configuration option `transport_lbtrm_nak_generation_interval`; default 10,000ms). Note: Receivers for these messages' topics will also report related messages as unrecoverable, with LBM\_MSG\_UNRECOVERABLE\_LOSS for an individual message and LBM\_MSG\_UNRECOVERABLE\_LOSS\_BURST for a burst loss event. However, it is possible for these application-level message declarations to occur even without increments to this counter, as the transport is unaware of the topic content of messages and may still be trying to deliver related lost packets.

**7.38.1.19 `lbtm_ulong_t lbtm_rev_transport_stats_lbtrm_t_stct::lbtm_msgs_rcved`**

Number of messages or message fragments received over an LBT-RM transport. A single datagram may contain one or more messages or a fragment of a larger message. For fragmented messages larger than configuration option `transport_lbtrm_datagram_max_size` (default 8KB), this count reflects the number of datagrams used to constitute those messages. Thus, this number is equal to or greater than the datagram counter (`msgs_rcved`, above). This number also includes messages received for which there was no interested receiver, which is tallied in the `lbtm_msgs_no_topic_rcved` counter (below).

**7.38.1.20 `lbtm_ulong_t lbtm_rev_transport_stats_lbtrm_t_stct::lbtm_msgs_no_topic_rcved`**

Number of messages received that were not for a topic of interest to the receiver. A high value (relative to, or approaching `lbtm_msgs_rcved` above) indicates more CPU time required to filter out uninteresting topics, in which case, consider reconfiguring sources to filter more aggressively at the transport layer.

**7.38.1.21 `lbtm_ULONG_t lbtm_rcv_transport_stats_lbtrm_t_stct::lbtm_reqs_rcved`**

Number of UM request messages received (message type LBM\_MSG\_REQUEST).

**7.38.1.22 `lbtm_ULONG_t lbtm_rcv_transport_stats_lbtrm_t_stct::dgrams_dropped_size`**

Number of datagrams discarded due to being smaller than the size designated in the datagram's size field.

**7.38.1.23 `lbtm_ULONG_t lbtm_rcv_transport_stats_lbtrm_t_stct::dgrams_dropped_type`**

Number of datagrams discarded due to bad packet type. The datagram's type field must match the expectations of the receiver transport.

**7.38.1.24 `lbtm_ULONG_t lbtm_rcv_transport_stats_lbtrm_t_stct::dgrams_dropped_version`**

Number of datagrams discarded due to version mismatch. The datagram's version field must match the expectations of the receiver transport.

**7.38.1.25 `lbtm_ULONG_t lbtm_rcv_transport_stats_lbtrm_t_stct::dgrams_dropped_hdr`**

Number of datagrams discarded due to bad header type. These datagrams appeared to be intact, but with an unrecognizable header format.

**7.38.1.26 `lbtm_ULONG_t lbtm_rcv_transport_stats_lbtrm_t_stct::dgrams_dropped_other`**

Number of unrecognizable datagrams discarded due to reasons other than those determined by the above counts. They could be garbled, or possibly be from foreign or incompatible software at the other end.

**7.38.1.27 `lbtm_ULONG_t lbtm_rcv_transport_stats_lbtrm_t_stct::out_of_order`**

Number of out-of-order LBT-RM transport datagrams received. A datagram is counted as out of order if it fills a previously detected sequence gap, but is not a retransmission. Note that if the duplicates counter (duplicate\_data, above) increases along with this statistic, this implies the arrivals of retransmitted datagrams before their originals.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.39 `lbt_rx_transport_stats_lbtru_t_stct` Struct Reference

Structure that holds datagram statistics for receiver LBT-RU transports.

```
#include <lbt.h>
```

### Data Fields

- `lbt_ulong_t msgs_rcved`
- `lbt_ulong_t bytes_rcved`
- `lbt_ulong_t nak_pckts_sent`
- `lbt_ulong_t naks_sent`
- `lbt_ulong_t lost`
- `lbt_ulong_t ncfs_ignored`
- `lbt_ulong_t ncfs_shed`
- `lbt_ulong_t ncfs_rx_delay`
- `lbt_ulong_t ncfs_unknown`
- `lbt_ulong_t nak_stm_min`
- `lbt_ulong_t nak_stm_mean`
- `lbt_ulong_t nak_stm_max`
- `lbt_ulong_t nak_tx_min`
- `lbt_ulong_t nak_tx_mean`
- `lbt_ulong_t nak_tx_max`
- `lbt_ulong_t duplicate_data`
- `lbt_ulong_t unrecovered_txw`
- `lbt_ulong_t unrecovered_tmo`
- `lbt_ulong_t lbt_msgs_rcved`
- `lbt_ulong_t lbt_msgs_no_topic_rcved`
- `lbt_ulong_t lbt_reqs_rcved`
- `lbt_ulong_t dgrams_dropped_size`
- `lbt_ulong_t dgrams_dropped_type`
- `lbt_ulong_t dgrams_dropped_version`
- `lbt_ulong_t dgrams_dropped_hdr`
- `lbt_ulong_t dgrams_dropped_sid`
- `lbt_ulong_t dgrams_dropped_other`

### 7.39.1 Field Documentation

#### 7.39.1.1 `lbt_ulong_t lbt_rcv_transport_stats_lbtru_t_stct::msgs_rcved`

Number of LBT-RU datagrams received. Depending on batching settings, a single LBT-RU datagram may contain one or more messages, or a fragment of a larger message. With LBT-RU, larger messages are split into fragment sizes limited by configuration option `transport_lbtru_datagram_max_size` (default 8KB).

#### 7.39.1.2 `lbt_ulong_t lbt_rcv_transport_stats_lbtru_t_stct::bytes_rcved`

Number of LBT-RU datagram bytes received, i.e., the total of lengths of all LBT-RU packets including UM header information.

#### 7.39.1.3 `lbt_ulong_t lbt_rcv_transport_stats_lbtru_t_stct::nak_pkts_sent`

Number of NAK packets sent by the receiver transport. UM batches NAKs into NAK packets to save network bandwidth. This should always be less than or equal to the number of individual NAKs sent (`naks_sent`, below).

#### 7.39.1.4 `lbt_ulong_t lbt_rcv_transport_stats_lbtru_t_stct::naks_sent`

Number of individual NAKs sent by the receiver transport. This may differ from the tally of lost datagrams (below) due to reasons such as

- Other receiver transports may have already sent a NAK for the same lost datagram, resulting in a retransmitted lost datagram (or an NCF) to arrive at this receiver transport before it has a chance to issue a NAK, or
- During periods of heavy loss, receiver transports may be forced to issue multiple NAKs per lost datagram (controlled by configuration options `transport_lbtru_nak_generation_interval` and `transport_lbtru_nak_backoff_interval`) until either the retransmission is received or the datagram is declared unrecovered (which may ultimately lead to UM delivering an `LBM_MSG_UNRECOVERABLE_LOSS` notification to the receiver application).

#### 7.39.1.5 `lbt_ulong_t lbt_rcv_transport_stats_lbtru_t_stct::lost`

Number of LBT-RU datagrams detected as lost.

#### 7.39.1.6 `lbtu_ncfs_ignored`

Number of NCFs received from a source transport with reason code "ignored". If a source transport receives a NAK for a datagram that it has recently retransmitted, it sends an "NCF ignored" and does not retransmit. How "recently" is determined by the configuration option `source_transport_lbtru_ignore_interval` (default 500ms). If this count is high, a receiver transport may be having trouble receiving retransmissions, or the ignore interval may be set too long.

#### 7.39.1.7 `lbtu_ncfs_shed`

Number of NCFs received with reason code "shed". When a source transport's retransmit queue and rate limiter are both at maximum, it responds to a NAK by sending an "NCF shed", and does not retransmit. The receiver transport should wait, then send another NAK. If this count is high, one or more crybaby receiver transports may be clogging the source transport's retransmit queue.

#### 7.39.1.8 `lbtu_ncfs_rx_delay`

Number of NCFs received with reason code "rx\_delay". When a source transport's retransmit rate limiter prevents it from immediately retransmitting any more lost datagrams, it responds to a NAK by sending an "NCF rx\_delay", then queues the retransmission for a later send. The receiver transport should wait for the retransmission and not immediately send another NAK. If this count is high, one or more crybaby receiver transports may be clogging the source transport's retransmit queue.

#### 7.39.1.9 `lbtu_ncfs_unknown`

Number of NCFs received with reason code "unknown". These are NCFs with a reason code this receiver transport does not recognize. After a delay (set by configuration option `transport_lbtru_nak_suppress_interval` (default 1000ms), it resends the NAK. This counter should never be greater than 0 unless applications linked with different versions of Ultra Messaging software coexist on the same network.

#### 7.39.1.10 `lbtu_nak_stm_min`

Minimum time (in milliseconds), i.e., the shortest time recorded so far for a lost message to be recovered. If this time is greater than configuration option `transport_lbtru_nak_backoff_interval`, it may be taking multiple NAKs to initiate retransmissions, indicating a lossy network.

**7.39.1.11 `lbt_ulong_t lbt_rx_transport_stats_lbtru_t_stct::nak_stm_mean`**

Mean time (in milliseconds) in which loss recovery was accomplished. This is an exponentially weighted moving average (weighted to more recent) for accumulated measured recovery times. Ideally this field should be as close to your minimum recovery time (`nak_stm_min`, above) as possible. High mean recovery times indicate a lossy network.

**7.39.1.12 `lbt_ulong_t lbt_rx_transport_stats_lbtru_t_stct::nak_stm_max`**

Maximum time (in milliseconds), i.e., the longest time recorded so far for a lost message to be recovered. If this time is near or equal to the configuration option `transport_lbtru_nak_generation_interval` setting, you have likely experienced some level of unrecoverable loss.

**7.39.1.13 `lbt_ulong_t lbt_rx_transport_stats_lbtru_t_stct::nak_tx_min`**

Minimum number of times per lost message that a receiver transport transmitted a NAK, i.e., the lowest value collected so far. A value greater than 1 indicates a chronically lossy network.

**7.39.1.14 `lbt_ulong_t lbt_rx_transport_stats_lbtru_t_stct::nak_tx_mean`**

Mean number of times per lost message that a receiver transport transmitted a NAK. Ideally this should be at or near 1. A higher value indicates a lossy network. This is an exponentially weighted moving average (weighted to more recent) for accumulated NAKs per lost message.

**7.39.1.15 `lbt_ulong_t lbt_rx_transport_stats_lbtru_t_stct::nak_tx_max`**

Maximum number of times per lost message that a receiver transport transmitted a NAK, i.e., the highest value collected so far. A value higher than 1 suggests that there may have been some unrecoverable loss on the network during the sample period. A significantly high value (compared to the mean number) implies an isolated incident.

**7.39.1.16 `lbt_ulong_t lbt_rx_transport_stats_lbtru_t_stct::duplicate_data`**

Number of duplicate LBT-RU datagrams received. A large number can indicate a lossy network, primarily due to other receiver transports requesting retransmissions that this receiver transport has already successfully received. Such duplicates require extra effort for filtering, and this should be investigated.

**7.39.1.17 `lbtu_txw`**

Number of LBT-RU datagrams unrecovered (LBM\_MSG\_UNRECOVERABLE\_LOSS delivered to receiver application) due to transmission window advance. This means that the message was no longer in the source-side transmission window and therefore not retransmitted. The window size is set by transport configuration option `lbtrt_transmission_window_size` (default 24MB).

**7.39.1.18 `lbtu_tmo`**

Number of LBT-RU datagrams unrecovered due to a retransmission not received within the NAK generation interval (set by configuration option `transport_lbtrm_nak_generation_interval`; default 10,000ms). Note: Receivers for these messages' topics will also report related messages as unrecoverable, with `LBM_MSG_UNRECOVERABLE_LOSS` for an individual message and `LBM_MSG_UNRECOVERABLE_LOSS_BURST` for a burst loss event. However, it is possible for these application-level message declarations to occur even without increments to this counter, as the transport is unaware of the topic content of messages and may still be trying to deliver related lost packets.

**7.39.1.19 `lbtu_rcved`**

Number of messages or message fragments received over an LBT-RU transport. A single datagram may contain one or more messages or a fragment of a larger message. For fragmented messages larger than configuration option `transport_lbtrt_datagram_max_size` (default 8KB), this count reflects the number of datagrams used to constitute those messages. Thus, this number is equal to or greater than the datagram counter (`msgs_rcved`, above). This number also includes messages received for which there was no interested receiver, which is tallied in the `lbtu_no_topic_rcved` counter (below).

**7.39.1.20 `lbtu_no_topic_rcved`**

Number of messages received that were not for a topic of interest to the receiver. A high value (relative to, or approaching `lbtu_rcved` above) indicates more CPU time required to filter out uninteresting topics, in which case, consider reconfiguring sources to filter more aggressively at the transport layer.

**7.39.1.21 `lbtu_reqs_rcved`**

Number of UM request messages received (message type `LBM_MSG_REQUEST`).

**7.39.1.22 `lbtm_ulong_t lbtm_rcv_transport_stats_lbtru_t_stct::dgrams_dropped_size`**

Number of datagrams discarded due to being smaller than the size designated in the datagram's size field.

**7.39.1.23 `lbtm_ulong_t lbtm_rcv_transport_stats_lbtru_t_stct::dgrams_dropped_type`**

Number of datagrams discarded due to bad packet type. The datagram's type field must match the expectations of the receiver transport.

**7.39.1.24 `lbtm_ulong_t lbtm_rcv_transport_stats_lbtru_t_stct::dgrams_dropped_version`**

Number of datagrams discarded due to version mismatch. The datagram's version field must match the expectations of the receiver transport.

**7.39.1.25 `lbtm_ulong_t lbtm_rcv_transport_stats_lbtru_t_stct::dgrams_dropped_hdr`**

Number of datagrams discarded due to bad header type. These datagrams appeared to be intact, but with an unrecognizable header format.

**7.39.1.26 `lbtm_ulong_t lbtm_rcv_transport_stats_lbtru_t_stct::dgrams_dropped_sid`**

Number of datagrams discarded due to session ID mismatch. These datagrams appeared to be correctly formed, but with an unmatched/unrecognized session ID field.

**7.39.1.27 `lbtm_ulong_t lbtm_rcv_transport_stats_lbtru_t_stct::dgrams_dropped_other`**

Number of unrecognizable datagrams discarded due to reasons other than those determined by the above counts. They could be garbled, or possibly be from foreign or incompatible software at the other end.

The documentation for this struct was generated from the following file:

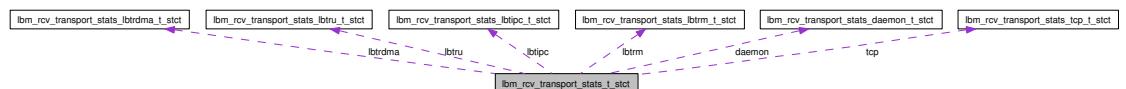
- [lbtm.h](#)

## 7.40 lbm\_rcv\_transport\_stats\_t\_stct Struct Reference

Structure that holds statistics for receiver transports.

```
#include <lbm.h>
```

Collaboration diagram for lbm\_rcv\_transport\_stats\_t\_stct:



### Data Fields

- int `type`
- char `source` [LBM\_MSG\_MAX\_SOURCE\_LEN]
- union {
  - `lbm_rcv_transport_stats_tcp_t tcp`
  - `lbm_rcv_transport_stats_lbtrm_t lbtrm`
  - `lbm_rcv_transport_stats_daemon_t daemon`
  - `lbm_rcv_transport_stats_lbtru_t lbtru`
  - `lbm_rcv_transport_stats_lbtrpc_t lbtrpc`
  - `lbm_rcv_transport_stats_lbtrdma_t lbtrdma`
} **transport**
- char `_fill` [LBM\_EXTERNAL\_STRUCT\_FILL\_SIZE]

#### 7.40.1 Detailed Description

This structure holds statistics for all receiver transports. The structure is filled in when statistics for receiver transports are requested.

#### 7.40.2 Field Documentation

##### 7.40.2.1 int lbm\_rcv\_transport\_stats\_t\_stct::type

Type of transport (e.g., LBM\_TRANSPORT\_STAT\_TCP, LBM\_TRANSPORT\_STAT\_LBTRM, etc.).

**7.40.2.2 `char lbt_recv_transport_stats_t_stct::source[LBM_MSG_MAX_SOURCE_LEN]`**

Source string of transport session, the format of which depends on the transport type.  
For string formats and examples, see [lbt\\_transport\\_source\\_info\\_t\\_stct](#).

**7.40.2.3 `lbt_recv_transport_stats_tcp_t lbt_recv_transport_stats_t_stct::tcp`**

The statistics for receiver TCP transports.

**7.40.2.4 `lbt_recv_transport_stats_lbtrm_t lbt_recv_transport_stats_t_stct::lbtrm`**

The statistics for receiver LBT-RM transports.

**7.40.2.5 `lbt_recv_transport_stats_daemon_t lbt_recv_transport_stats_t_stct::daemon`**

These statistics have been deprecated.

**7.40.2.6 `lbt_recv_transport_stats_lbtru_t lbt_recv_transport_stats_t_stct::lbtru`**

The statistics for receiver LBT-RU transports.

**7.40.2.7 `lbt_recv_transport_stats_lbtipc_t lbt_recv_transport_stats_t_stct::lbtipc`**

The statistics for receiver LBT-IPC transports.

**7.40.2.8 `lbt_recv_transport_stats_lbtrdma_t lbt_recv_transport_stats_t_stct::lbtrdma`**

The statistics for receiver LBT-RDMA transports.

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.41 **lbm\_rcv\_transport\_stats\_tcp\_t\_stct** Struct Reference

Structure that holds datagram statistics for receiver TCP transports.

```
#include <lbm.h>
```

### Data Fields

- `lbm_ulong_t bytes_rcved`
- `lbm_ulong_t lbm_msgs_rcved`
- `lbm_ulong_t lbm_msgs_no_topic_rcved`
- `lbm_ulong_t lbm_reqs_rcved`

#### 7.41.1 Field Documentation

##### 7.41.1.1 `lbm_ulong_t lbm_rcv_transport_stats_tcp_t_stct::bytes_rcved`

Number of TCP datagram bytes received, i.e., the total of lengths of all TCP packets including UM header information.

##### 7.41.1.2 `lbm_ulong_t lbm_rcv_transport_stats_tcp_t_stct::lbm_msgs_rcved`

Number of messages or message fragments received over a TCP transport. A single datagram may contain one or more messages or a fragment of a larger message. For fragmented messages larger than configuration option `transport_tcp_datagram_max_size` (default 64KB), this count reflects the number of datagrams used to constitute those messages. Thus, this number is equal to or greater than the datagram counter (`msgs_rcved`, above). This number also includes messages received for which there was no interested receiver, which is tallied in the `lbm_msgs_no_topic_rcved` counter (below).

##### 7.41.1.3 `lbm_ulong_t lbm_rcv_transport_stats_tcp_t_stct::lbm_msgs_no_topic_rcved`

Number of messages received that were not for a topic of interest to the receiver. A high value (relative to, or approaching `lbm_msgs_rcved` above) indicates more CPU time required to filter out uninteresting topics, in which case, consider reconfiguring sources to filter more aggressively at the transport layer.

**7.41.1.4    lbtm\_ulong\_t lbtm\_rcv\_transport\_stats\_tcp\_t\_stct::lbtm\_reqs\_rcved**

Number of UM request messages received (message type LBM\_MSG\_REQUEST).

The documentation for this struct was generated from the following file:

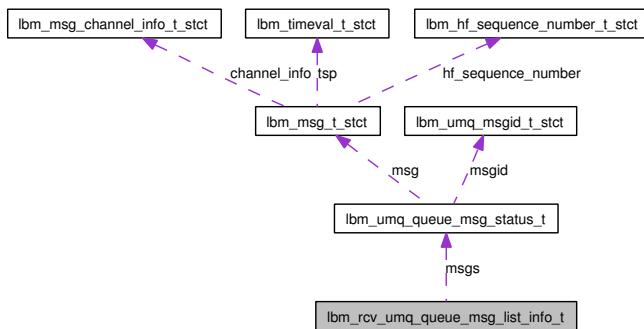
- [lbtm.h](#)

## 7.42 lbm\_rcv\_umq\_queue\_msg\_list\_info\_t Struct Reference

Struct containing an array of UMQ messages listed via `lbm_rcv_umq_queue_msg_list`.

```
#include <lbm.h>
```

Collaboration diagram for `lbm_rcv_umq_queue_msg_list_info_t`:



### Data Fields

- `lbm_umq_queue_msg_status_t * msgs`
- `lbm_uint64_t num_msgs`

#### 7.42.1 Field Documentation

##### 7.42.1.1 `lbm_umq_queue_msg_status_t* lbm_rcv_umq_queue_msg_list_info_t::msgs`

An array of queued messages.

##### 7.42.1.2 `lbm_uint64_t lbm_rcv_umq_queue_msg_list_info_t::num_msgs`

The length of the queued messages array.

The documentation for this struct was generated from the following file:

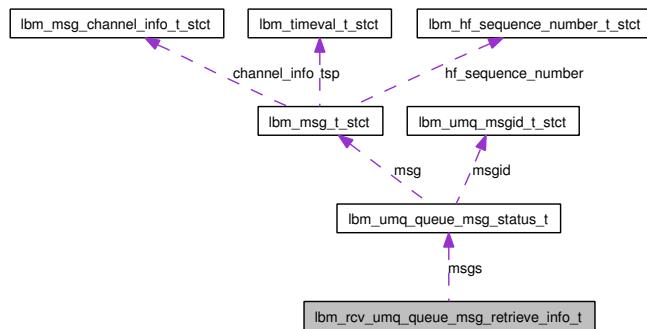
- `lbm.h`

## 7.43 `lbt_rcv_umq_queue_msg_retrieve_info_t` Struct Reference

Struct containing an array of UMQ messages retrieved via `lbt_rcv_umq_queue_msg_retrieve`.

```
#include <lbt.h>
```

Collaboration diagram for `lbt_rcv_umq_queue_msg_retrieve_info_t`:



### Data Fields

- `lbt_umq_queue_msg_status_t * msgs`
- int `num_msgs`

#### 7.43.1 Field Documentation

##### 7.43.1.1 `lbt_umq_queue_msg_status_t* lbt_rcv_umq_queue_msg_retrieve_info_t::msgs`

An array of the retrieved messages.

##### 7.43.1.2 `int lbt_rcv_umq_queue_msg_retrieve_info_t::num_msgs`

The length of the retrieved message array.

The documentation for this struct was generated from the following file:

- `lbt.h`

## 7.44 lbm\_serialized\_response\_t\_stct Struct Reference

Structure that holds a serialized UM response object.

```
#include <lbm.h>
```

### Data Fields

- char **serial\_response** [32]

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.45 **lbt\_src\_cost\_func\_t\_stct** Struct Reference

Structure that holds the "source\_cost\_evaluation\_function" context attribute.

```
#include <lbt.h>
```

### Data Fields

- **lbt\_src\_cost\_function\_cb** **cost\_cb**
- **void \*** **clientd**

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.46 `lbt_src_event_flight_size_notification_t_stct` Struct Reference

Structure that holds flight size notification event data.

```
#include <lbt.h>
```

### Data Fields

- int `type`
- int `state`

#### 7.46.1 Detailed Description

A structure used to indicate a state change in flight size status

#### 7.46.2 Field Documentation

##### 7.46.2.1 int `lbt_src_event_flight_size_notification_t_stct::type`

Specifies which flight size's state changed

##### 7.46.2.2 int `lbt_src_event_flight_size_notification_t_stct::state`

Current state of specified flight size

The documentation for this struct was generated from the following file:

- `lbt.h`

## 7.47 `lbt_src_event_sequence_number_info_t_stct` Struct Reference

Structure that holds sequence number information for a message sent by a source.

```
#include <lbt.h>
```

### Data Fields

- int `flags`
- `lbt_uint_t first_sequence_number`
- `lbt_uint_t last_sequence_number`
- void \* `msg_clientd`

#### 7.47.1 Detailed Description

A structure used with UM sources that informs the application the sequence numbers used with a message.

See also:

[lbt\\_src\\_send\\_ex](#)

#### 7.47.2 Field Documentation

##### 7.47.2.1 int `lbt_src_event_sequence_number_info_t_stct::flags`

Flags that indicate which optional portions are included

##### 7.47.2.2 `lbt_uint_t lbt_src_event_sequence_number_info_t_stct::first_sequence_number`

First sequence number for the message set

##### 7.47.2.3 `lbt_uint_t lbt_src_event_sequence_number_info_t_stct::last_sequence_number`

Last sequence number for the message set

**7.47.2.4 void\* lbm\_src\_event\_sequence\_number\_info\_t\_stct::msg\_clientd**

The clientd pointer passed in for the message

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.48 `lbt_src_event_ume_ack_ex_info_t_stct` Struct Reference

Structure that holds ACK information for a given message in an extended form.

```
#include <lbt.h>
```

### Data Fields

- int `flags`
- `lbt_uint_t sequence_number`
- `lbt_uint_t rev_registration_id`
- char `store` [LBM\_UME\_MAX\_STORE\_STRLEN]
- void \* `msg_clientd`
- `lbt_ushort_t store_index`

#### 7.48.1 Detailed Description

A structure used with UMP sources to indicate message acknowledgment by the store and UMP receivers (extended form).

#### 7.48.2 Field Documentation

##### 7.48.2.1 int `lbt_src_event_ume_ack_ex_info_t_stct::flags`

Flags

##### 7.48.2.2 `lbt_uint_t lbt_src_event_ume_ack_ex_info_t_stct::sequence_number`

The sequence number of the message that is being acknowledged

##### 7.48.2.3 `lbt_uint_t lbt_src_event_ume_ack_ex_info_t_stct::rev_registration_id`

The registration ID for the receiver. This field is 0 (zero) if using the structure for a message acknowledgment event.

**7.48.2.4 `char lbm_src_event_ume_ack_ex_info_t_stct::store[LBM_UME_-MAX_STORE_STRLEN]`**

The store involved. This field is 0 (zero) if the acknowledgement indicates the message is stable at a quorum of stores or if using the structure for a delivery confirmation event.

**7.48.2.5 `void* lbm_src_event_ume_ack_ex_info_t_stct::msg_clientd`**

The clientd pointer passed in for the message

**7.48.2.6 `lbm_ushort_t lbm_src_event_ume_ack_ex_info_t_stct::store_index`**

The store index of the store involved. This field is 0 (zero) if the acknowledgement indicates the message is stable at a quorum of stores or if using the structure for a delivery confirmation event.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.49 `lbt_src_event_ume_ack_info_t_stct` Struct Reference

Structure that holds ACK information for a given message.

```
#include <lbt.h>
```

### Data Fields

- `lbt_uint_t sequence_number`
- `lbt_uint_t rcv_registration_id`
- `void * msg_clientd`

#### 7.49.1 Detailed Description

A structure used with UMP sources to indicate message acknowledgment by the store and UMP receivers.

#### 7.49.2 Field Documentation

##### 7.49.2.1 `lbt_uint_t lbt_src_event_ume_ack_info_t_stct::sequence_number`

The sequence number of the message that is being acknowledged

##### 7.49.2.2 `lbt_uint_t lbt_src_event_ume_ack_info_t_stct::rcv_registration_id`

The registration ID for the receiver

##### 7.49.2.3 `void* lbt_src_event_ume_ack_info_t_stct::msg_clientd`

The clientd pointer passed in for the message

The documentation for this struct was generated from the following file:

- `lbt.h`

## 7.50 `lbt_src_event_ume_deregistration_ex_t_stct` Struct Reference

Structure that holds store deregistration information for the UMP source in an extended form.

```
#include <lbt.h>
```

### Data Fields

- int `flags`
- lbt\_uint32\_t `registration_id`
- lbt\_uint\_t `sequence_number`
- lbt\_ushort\_t `store_index`
- char `store` [LBM\_UME\_MAX\_STORE\_STRLEN]

#### 7.50.1 Detailed Description

A structure used with UMP sources to indicate successful deregistration (extended form).

#### 7.50.2 Field Documentation

##### 7.50.2.1 int `lbt_src_event_ume_deregistration_ex_t_stct::flags`

Flags

##### 7.50.2.2 lbt\_uint32\_t `lbt_src_event_ume_deregistration_ex_t_stct::registration_id`

The registration ID for the source

##### 7.50.2.3 lbt\_uint\_t `lbt_src_event_ume_deregistration_ex_t_stct::sequence_number`

The sequence number of the last message stored for the source as reported by the store

**7.50.2.4   **lbm\_ushort\_t** **lbm\_src\_event\_ume\_deregistration\_ex\_t\_stct::store\_index****

The store index of the store involved.

**7.50.2.5   **char** **lbm\_src\_event\_ume\_deregistration\_ex\_t\_stct::store[LBM\_UME\_MAX\_STORE\_STRLEN]****

The store that was registered with.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.51 `lbt_src_event_ume_registration_complete_ex_t_stct` Struct Reference

Structure that holds information for sources after registration is complete to all involved stores.

```
#include <lbt.h>
```

### Data Fields

- int `flags`
- lbt\_uint\_t `sequence_number`

#### 7.51.1 Detailed Description

A structure used with UMP sources to indicate successful registration to quorum or to all stores involved.

#### 7.51.2 Field Documentation

##### 7.51.2.1 int `lbt_src_event_ume_registration_complete_ex_t_stct::flags`

Flags

##### 7.51.2.2 lbt\_uint\_t `lbt_src_event_ume_registration_complete_ex_t_stct::sequence_number`

The sequence number that will be the first sent

The documentation for this struct was generated from the following file:

- `lbt.h`

## 7.52 `lbt_src_event_ume_registration_ex_t_stct` Struct Reference

Structure that holds store registration information for the UMP source in an extended form.

```
#include <lbt.h>
```

### Data Fields

- int `flags`
- `lbt_uint_t registration_id`
- `lbt_uint_t sequence_number`
- `lbt_ushort_t store_index`
- char `store` [LBM\_UME\_MAX\_STORE\_STRLEN]

#### 7.52.1 Detailed Description

A structure used with UMP sources to indicate successful registration (extended form).

#### 7.52.2 Field Documentation

##### 7.52.2.1 int `lbt_src_event_ume_registration_ex_t_stct::flags`

Flags

##### 7.52.2.2 `lbt_uint_t lbt_src_event_ume_registration_ex_t_stct::registration_id`

The registration ID for the source

##### 7.52.2.3 `lbt_uint_t lbt_src_event_ume_registration_ex_t_stct::sequence_number`

The sequence number for the source to use as reported by the store

##### 7.52.2.4 `lbt_ushort_t lbt_src_event_ume_registration_ex_t_stct::store_index`

The store index of the store involved

**7.52.2.5 char `lbt_src_event_ume_registration_ex_t::store[LBM_UME_-  
MAX_STORE_STRLEN]`**

The store that was registered with

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.53 `lbt_src_event_ume_registration_t_stct` Struct Reference

Structure that holds store registration information for the UMP source.

```
#include <lbt.h>
```

### Data Fields

- `lbt_uint_t registration_id`

#### 7.53.1 Detailed Description

A structure used with UMP sources to indicate successful registration.

#### 7.53.2 Field Documentation

##### 7.53.2.1 `lbt_uint_t lbt_src_event_ume_registration_t_stct::registration_id`

The registration ID for the source

The documentation for this struct was generated from the following file:

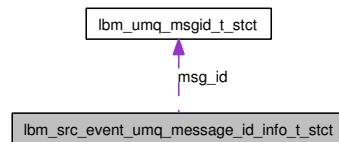
- [lbt.h](#)

## 7.54 `lbt_src_event_umq_message_id_info_t_stct` Struct Reference

Structure that holds Message ID information for a message sent by a sending UMQ application.

```
#include <lbt.h>
```

Collaboration diagram for `lbt_src_event_umq_message_id_info_t_stct`:



### Data Fields

- int `flags`
- `lbt_umq_msgid_t msg_id`
- void \* `msg_clientd`

#### 7.54.1 Detailed Description

See also:

[lbt\\_src\\_send\\_ex](#) A structure used with UMQ sending applications that informs the application of the UMQ Message ID used with a message.

#### 7.54.2 Field Documentation

##### 7.54.2.1 int `lbt_src_event_umq_message_id_info_t_stct::flags`

Flags that indicate which optional portions are included

##### 7.54.2.2 `lbt_umq_msgid_t lbt_src_event_umq_message_id_info_t_stct::msg_id`

Message ID for the message sent

##### 7.54.2.3 `void* lbt_src_event_umq_message_id_info_t_stct::msg_clientd`

The clientd pointer passed in for the message

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.55 `lbm_src_event_umq_registration_complete_ex_t_stct` Struct Reference

Structure that holds information for sources after registration is complete to all involved queue instances.

```
#include <lbm.h>
```

### Data Fields

- int `flags`
- `lbm_uint_t queue_id`
- char `queue` [LBM\_UMQ\_MAX\_QUEUE\_STRLEN]

#### 7.55.1 Detailed Description

A structure used with UMQ sources to indicate successful source registration to quorum or to all queue instances involved.

#### 7.55.2 Field Documentation

##### 7.55.2.1 `int lbm_src_event_umq_registration_complete_ex_t_stct::flags`

Flags that indicate which optional portions are included

##### 7.55.2.2 `lbm_uint_t lbm_src_event_umq_registration_complete_ex_t_stct::queue_id`

The Queue ID of the queue

##### 7.55.2.3 `char lbm_src_event_umq_registration_complete_ex_t_stct::queue[LBM_UMQ_MAX_QUEUE_STRLEN]`

The name of the queue registered with

The documentation for this struct was generated from the following file:

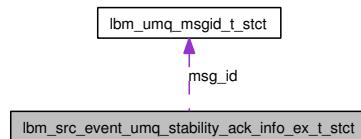
- [lbm.h](#)

## 7.56 `lbt_src_event_umq_stability_ack_info_ex_t_stct` Struct Reference

Structure that holds UMQ ACK information for a given message in an extended form.

```
#include <lbt.h>
```

Collaboration diagram for `lbt_src_event_umq_stability_ack_info_ex_t_stct`:



### Data Fields

- int `flags`
- `lbt_umq_msid_t msg_id`
- `lbt_uint_t first_sequence_number`
- `lbt_uint_t last_sequence_number`
- `lbt_uint_t queue_id`
- `lbt_uint_t queue_instance_index`
- `void * msg_clientd`
- `char queue_instance [LBM_UME_MAX_STORE_STRLEN]`
- `char queue [LBM_UML_MAX_QUEUE_STRLEN]`

#### 7.56.1 Detailed Description

A structure used with UMQ source applications to indicate message acknowledgment by a queue instance.

#### 7.56.2 Field Documentation

##### 7.56.2.1 int `lbt_src_event_umq_stability_ack_info_ex_t_stct::flags`

Flags that indicate which optional portions are included

##### 7.56.2.2 `lbt_umq_msid_t lbt_src_event_umq_stability_ack_info_ex_t_stct::msg_id`

Message ID of the message being acknowledged

**7.56.2.3 `lbt_uint_t lbt_src_event_umq_stability_ack_info_ex_t_stct::first_sequence_number`**

First sequence number for the message after being fragmented

**7.56.2.4 `lbt_uint_t lbt_src_event_umq_stability_ack_info_ex_t_stct::last_sequence_number`**

Last sequence number for the message after being fragmented

**7.56.2.5 `lbt_uint_t lbt_src_event_umq_stability_ack_info_ex_t_stct::queue_id`**

The Queue ID of the queue

**7.56.2.6 `lbt_uint_t lbt_src_event_umq_stability_ack_info_ex_t_stct::queue_instance_index`**

The index of the instance of the queue acknowledging the message

**7.56.2.7 `void* lbt_src_event_umq_stability_ack_info_ex_t_stct::msg_clientd`**

The clientd pointer passed in for the message

**7.56.2.8 `char lbt_src_event_umq_stability_ack_info_ex_t_stct::queue_instance[LBM_UME_MAX_STORE_STRLEN]`**

The instance of the queue acknowledging the message

**7.56.2.9 `char lbt_src_event_umq_stability_ack_info_ex_t_stct::queue_name[LBM_UMQ_MAX_QUEUE_STRLEN]`**

The name of the queue

The documentation for this struct was generated from the following file:

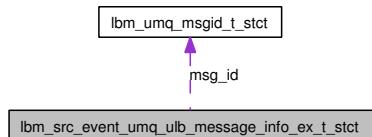
- [lbt.h](#)

## 7.57 `lbt_src_event_umq_ulb_message_info_ex_t_stct` Struct Reference

Structure that holds UMQ ULB message information in an extended form.

```
#include <lbt.h>
```

Collaboration diagram for `lbt_src_event_umq_ulb_message_info_ex_t_stct`:



### Data Fields

- int `flags`
- `lbt_umq_mgid_t msg_id`
- `lbt_umq_regid_t registration_id`
- `lbt_uint_t first_sequence_number`
- `lbt_uint_t last_sequence_number`
- `lbt_uint_t assignment_id`
- `lbt_uint_t application_set_index`
- void \* `msg_clientd`
- char `receiver` [LBM\_UMQ\_ULB\_MAX\_RECEIVER\_STRLEN]

#### 7.57.1 Detailed Description

A structure used with UMQ ULB source applications to indicate message events.

#### 7.57.2 Field Documentation

##### 7.57.2.1 int `lbt_src_event_umq_ulb_message_info_ex_t_stct::flags`

Flags that indicate which optional portions are included

##### 7.57.2.2 `lbt_umq_mgid_t lbt_src_event_umq_ulb_message_info_ex_t_stct::msg_id`

Message ID of the message

**7.57.2.3 `lbt_umq_regid_t lbt_src_event_umq_ulb_message_info_ex_t::stct::registration_id`**

The registration ID of the receiver

**7.57.2.4 `lbt_uint_t lbt_src_event_umq_ulb_message_info_ex_t::stct::first_sequence_number`**

First sequence number for the message after being fragmented

**7.57.2.5 `lbt_uint_t lbt_src_event_umq_ulb_message_info_ex_t::stct::last_sequence_number`**

Last sequence number for the message after being fragmented

**7.57.2.6 `lbt_uint_t lbt_src_event_umq_ulb_message_info_ex_t::stct::assignment_id`**

The Assignment ID of the receiver

**7.57.2.7 `lbt_uint_t lbt_src_event_umq_ulb_message_info_ex_t::stct::application_set_index`**

The Application Set Index the receiver is in

**7.57.2.8 `void* lbt_src_event_umq_ulb_message_info_ex_t::stct::msg_clientd`**

The clientd pointer passed in for the message

**7.57.2.9 `char lbt_src_event_umq_ulb_message_info_ex_t::stct::receiver[LBM_UMQ_ULB_MAX_RECEIVER_STRLEN]`**

The receivers immediate message target string

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.58 `lmb_src_event_umq_ulb_receiver_info_ex_t_stct` Struct Reference

Structure that holds UMQ ULB receiver information in an extended form.

```
#include <lmb.h>
```

### Data Fields

- int `flags`
- `lmb_umq_regid_t registration_id`
- `lmb_uint_t assignment_id`
- `lmb_uint_t application_set_index`
- char `receiver` [LBM\_UMQ\_ULB\_MAX\_RECEIVER\_STRLEN]

### 7.58.1 Detailed Description

A structure used with UMQ ULB source applications to indicate receiver events.

### 7.58.2 Field Documentation

#### 7.58.2.1 int `lmb_src_event_umq_ulb_receiver_info_ex_t_stct::flags`

Flags that indicate which optional portions are included

#### 7.58.2.2 `lmb_umq_regid_t lmb_src_event_umq_ulb_receiver_info_ex_t_stct::registration_id`

The registration ID of the receiver

#### 7.58.2.3 `lmb_uint_t lmb_src_event_umq_ulb_receiver_info_ex_t_stct::assignment_id`

The Assignment ID of the receiver

#### 7.58.2.4 `lmb_uint_t lmb_src_event_umq_ulb_receiver_info_ex_t_stct::application_set_index`

The Application Set Index the receiver is in

**7.58.2.5 char l<sub>b</sub>m\_src\_event\_umq\_ulp\_receiver\_info\_ex\_t\_-  
stct::receiver[LBM\_UMQ\_ULB\_MAX\_RECEIVER\_STRLEN]**

The receivers immediate message target string

The documentation for this struct was generated from the following file:

- [l<sub>b</sub>m.h](#)

## 7.59 `lbt_src_event_wakeup_t_stct` Struct Reference

Structure that holds source wakeup event data.

```
#include <lbt.h>
```

### Data Fields

- int `flags`

#### 7.59.1 Detailed Description

A structure used to indicate the type of source that is now unblocked.

#### 7.59.2 Field Documentation

##### 7.59.2.1 int `lbt_src_event_wakeup_t_stct::flags`

OR'd set of flags indicating which context-level sources are now unblocked. Can contain one or more of `LBT_SRC_EVENT_WAKEUP_FLAG_*`

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.60 `lbt_src_notify_func_t_stct` Struct Reference

Structure that holds the callback for source notifications.

```
#include <lbt.h>
```

### Data Fields

- `lbt_src_notify_function_cb` **notifyfunc**
- `void *` **clientd**

#### 7.60.1 Detailed Description

A structure used with options to set/get a specific callback information

The documentation for this struct was generated from the following file:

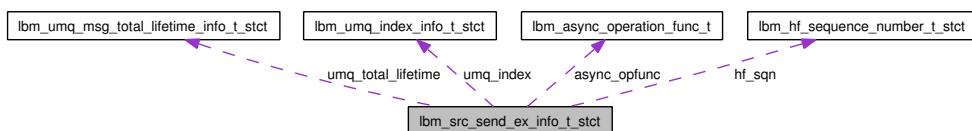
- [lbt.h](#)

## 7.61 `lbt_src_send_ex_info_t_stct` Struct Reference

Structure that holds information for the extended send calls A structure used with UM sources that utilize the extended send calls to pass options.

```
#include <lbt.h>
```

Collaboration diagram for `lbt_src_send_ex_info_t_stct`:



### Data Fields

- int `flags`
- void \* `ume_msg_clientd`
- `lbt_src_channel_info_t * channel_info`
- `lbt_apphdr_chain_t * apphdr_chain`
- `lbt_umq_index_info_t * umq_index`
- `lbt_umq_msg_total_lifetime_info_t * umq_total_lifetime`
- `lbt_msg_properties_t * properties`
- `lbt_hf_sequence_number_t hf_sqn`
- `lbt_async_operation_func_t * async_opfunc`

#### 7.61.1 Detailed Description

See also:

[lbt\\_src\\_send\\_ex](#)

#### 7.61.2 Field Documentation

##### 7.61.2.1 int `lbt_src_send_ex_info_t_stct::flags`

Flags that set which settings are active as defined by "LBM\_SRC\_SEND\_EX\_\*".  
Search [lbt.h](#) for LBM\_SRC\_SEND\_EX\_FLAG\_\*.

##### 7.61.2.2 void\* `lbt_src_send_ex_info_t_stct::ume_msg_clientd`

client data pointer to be passed back in source events for stability and confirmations

**7.61.2.3 `lbt_src_channel_info_t*` `lbt_src_send_ex_info_t_stct::channel_info`**

pointer to information used to send messages on a channel

**7.61.2.4 `lbt_apphdr_chain_t*` `lbt_src_send_ex_info_t_stct::apphdr_chain`**

pointer to information used to send messages using app header chains

**7.61.2.5 `lbt_umq_index_info_t*` `lbt_src_send_ex_info_t_stct::umq_index`**

pointer to information used to send messages and associate them with a given UMQ index

**7.61.2.6 `lbt_umq_msg_total_lifetime_info_t*` `lbt_src_send_ex_info_t_stct::umq_total_lifetime`**

pointer to information used to specify a message's total lifetime

**7.61.2.7 `lbt_msg_properties_t*` `lbt_src_send_ex_info_t_stct::properties`**

pointer to a message properties structure

**7.61.2.8 `lbt_hf_sequence_number_t` `lbt_src_send_ex_info_t_stct::hf_sqn`**

The hot failover sequence number to send

**7.61.2.9 `lbt_async_operation_func_t*` `lbt_src_send_ex_info_t_stct::async_opfunc`**

pointer to an asynchronous operation callback

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.62 `lbt_src_transport_stats_daemon_t_stct` Struct Reference

Structure that holds statistics for source daemon mode transport (deprecated).

```
#include <lbt.h>
```

### Data Fields

- `lbt_ulong_t bytes_buffered`

#### 7.62.1 Detailed Description

This structure holds statistics for source transports using the daemon mode. NOTE: daemon mode is deprecated and no longer available; this structure is retained for backward compatibility only.

#### 7.62.2 Field Documentation

##### 7.62.2.1 `lbt_ulong_t lbt_src_transport_stats_daemon_t_stct::bytes_buffered`

This statistic has been deprecated.

The documentation for this struct was generated from the following file:

- `lbt.h`

## 7.63 **lbtipc\_t\_stct** Struct Reference

Structure that holds datagram statistics for source LBT-IPC transports.

```
#include <lbtipc.h>
```

### Data Fields

- lbtipc\_t [num\\_clients](#)
- lbtipc\_t [msgs\\_sent](#)
- lbtipc\_t [bytes\\_sent](#)

#### 7.63.1 Field Documentation

##### 7.63.1.1 lbtipc\_t [lbtipc\\_t\\_stct::num\\_clients](#)

Number of receiver transports that are currently connected to this source transport.

##### 7.63.1.2 lbtipc\_t [lbtipc\\_t\\_stct::msgs\\_sent](#)

Number of LBT-IPC datagrams sent. Depending on batching settings, a single LBT-IPC datagram may contain one or more messages, or a fragment of a larger message. With LBT-IPC, larger messages are split into fragment sizes limited by configuration option `transport_lbtipc_datagram_max_size` (default 64KB).

##### 7.63.1.3 lbtipc\_t [lbtipc\\_t\\_stct::bytes\\_sent](#)

Number of LBT-IPC datagram bytes sent, i.e., the total of lengths of all LBT-IPC packets including UM header information.

The documentation for this struct was generated from the following file:

- [lbtipc.h](#)

## 7.64 `lbt_src_transport_stats_lbtrdma_t_stct` Struct Reference

Structure that holds datagram statistics for source LBT-RDMA transports.

```
#include <lbt.h>
```

### Data Fields

- `lbt_ulong_t num_clients`
- `lbt_ulong_t msgs_sent`
- `lbt_ulong_t bytes_sent`

#### 7.64.1 Field Documentation

##### 7.64.1.1 `lbt_ulong_t lbt_src_transport_stats_lbtrdma_t_stct::num_clients`

Number of receiver transports that are currently connected to this source transport.

##### 7.64.1.2 `lbt_ulong_t lbt_src_transport_stats_lbtrdma_t_stct::msgs_sent`

Number of LBT-RDMA datagrams sent. Depending on batching settings, a single LBT-RDMA datagram may contain one or more messages, or a fragment of a larger message. With LBT-RDMA, larger messages are split into fragment sizes limited by configuration option `transport_lbtrdma_datagram_max_size` (default 4KB).

##### 7.64.1.3 `lbt_ulong_t lbt_src_transport_stats_lbtrdma_t_stct::bytes_sent`

Number of LBT-RDMA datagram bytes sent, i.e., the total of lengths of all LBT-RDMA packets including UM header information.

The documentation for this struct was generated from the following file:

- `lbt.h`

## 7.65 `lbtm_src_transport_stats_lbtrm_t_stct` Struct Reference

Structure that holds datagram statistics for source LBT-RM transports.

```
#include <lbtm.h>
```

### Data Fields

- `lbtm_ulong_t msgs_sent`
- `lbtm_ulong_t bytes_sent`
- `lbtm_ulong_t txw_msgs`
- `lbtm_ulong_t txw_bytes`
- `lbtm_ulong_t nak_pkts_rcved`
- `lbtm_ulong_t naks_rcved`
- `lbtm_ulong_t naks_ignored`
- `lbtm_ulong_t naks_shed`
- `lbtm_ulong_t naks_rx_delay_ignored`
- `lbtm_ulong_t rxs_sent`
- `lbtm_ulong_t rctlr_data_msgs`
- `lbtm_ulong_t rctlr_rx_msgs`
- `lbtm_ulong_t rx_bytes_sent`

### 7.65.1 Field Documentation

#### 7.65.1.1 `lbtm_ulong_t lbtm_src_transport_stats_lbtrm_t_stct::msgs_sent`

Number of LBT-RM datagrams sent. Depending on batching settings, a single LBT-RM datagram may contain one or more messages, or a fragment of a larger message. With LBT-RM, larger messages are split into fragment sizes limited by configuration option `transport_lbtrm_datagram_max_size` (default 8KB).

#### 7.65.1.2 `lbtm_ulong_t lbtm_src_transport_stats_lbtrm_t_stct::bytes_sent`

Number of LBT-RM datagram bytes sent, i.e., the total of lengths of all LBT-RM packets including UM header information.

#### 7.65.1.3 `lbtm_ulong_t lbtm_src_transport_stats_lbtrm_t_stct::txw_msgs`

Number of LBT-RM datagrams currently in the transmission window. Each source transport session maintains a transmission window buffer (the size of which is set by

transport\_lbtrm\_transmission\_window\_size, default 24MB). When the source transport receives a NAK, the corresponding message for retransmission must be found in this transmission window.

#### 7.65.1.4 **lbt\_ulong\_t lbt\_src\_transport\_stats\_lbtrm\_t\_stct::txw\_bytes**

Number of bytes currently in the transmission window. See txw\_msgs (above) for a description of the transmission window. Typically, this count approaches its window size or exceeds it by a small amount.

#### 7.65.1.5 **lbt\_ulong\_t lbt\_src\_transport\_stats\_lbtrm\_t\_stct::nak\_pckts\_rcved**

Number of NAK packets received by this source transport. UM batches NAKs into NAK packets to save network bandwidth. This should always be less than or equal to naks\_rcved (below).

#### 7.65.1.6 **lbt\_ulong\_t lbt\_src\_transport\_stats\_lbtrm\_t\_stct::naks\_rcved**

Number of individual NAKs received by the source transport. When a source transport receives a NAK from a receiver transport, it may respond by re-transmitting the requested LBT-RM datagram, or it may send an NCF. The NAKing receiver transport responds to the NCF by waiting (timeout set by transport\_lbtrm\_nak\_suppress\_interval, default 1000 ms), then re-sending the NAK.

#### 7.65.1.7 **lbt\_ulong\_t lbt\_src\_transport\_stats\_lbtrm\_t\_stct::naks\_ignored**

Number of NAKs this source transport ignored and sent an NCF with reason code "ignored". A source transport ignores a NAK for a datagram it has already recently retransmitted. How "recently" is determined by the configuration option source transport\_lbtrm\_ignore\_interval (default 500ms). If this count is high, a receiver transport may be having trouble receiving retransmissions, or the ignore interval may be set too long.

#### 7.65.1.8 **lbt\_ulong\_t lbt\_src\_transport\_stats\_lbtrm\_t\_stct::naks\_shed**

Number of NAKs this source transport has shed by sending an NCF with reason code "shed". When a source transport's retransmit rate limiter and retransmit queue are both at maximum, it responds to a NAK by sending an "NCF shed", and does not retransmit. The receiver transport should wait, then send another NAK. If this count is high, one or more crybaby receiver transports may be clogging the source transport's retransmit queue.

#### 7.65.1.9 **lbtm\_ulong\_t lbtm\_src\_transport\_stats\_lbtrm\_t\_stct::naks\_rx\_delay\_ignored**

Number of NAKs this source transport has not yet processed because doing so would exceed its retransmit rate limit (set by configuration option `transport_lbtrm_retransmit_rate_limit`, default 5Mbps). For each of these NAKs, the source transport immediately sends an NFC rx\_delay, then queues the retransmission for a later send within the rate limit. If this count is high, one or more crybaby receiver transports may be clogging the source transport's retransmit queue.

#### 7.65.1.10 **lbtm\_ulong\_t lbtm\_src\_transport\_stats\_lbtrm\_t\_stct::rxs\_sent**

Number of LBT-RM datagrams retransmitted by this source transport (incremented under the same circumstances as `rx_bytes_sent`, below). In a normal, light-loss scenario, most NAKs induce a retransmission. When losses becomes heavy and/or many receiver transports begin losing the same LBT-RM datagrams, NCF-related no-retransmit counts (`naks_ignored`, `naks_shed` and `naks_rx_delay_ignored`) may begin to inflate, and retransmissions (`rxs_sent/rx_bytes_sent` counts) may become significantly lower than NAKS received (`naks_rcved`).

#### 7.65.1.11 **lbtm\_ulong\_t lbtm\_src\_transport\_stats\_lbtrm\_t\_stct::rctlr\_data\_msgs**

Number of LBT-RM datagrams currently queued by the data rate limiter. When a source transport attempts to send messages (any type) faster than its data rate limiter allows (set by configuration option `transport_lbtrm_data_rate_limit`, default 10Mbps), the data rate limiter queues the messages until they can be sent within the data rate limit.

#### 7.65.1.12 **lbtm\_ulong\_t lbtm\_src\_transport\_stats\_lbtrm\_t\_stct::rctlr\_rx\_msgs**

Number of LBT-RM transport retransmission datagrams currently queued by the retransmit rate limiter. When a source transport attempts to send retransmissions faster than its retransmit rate limiter allows (set by configuration option `transport_lbtrm_retransmit_rate_limit`, default 5Mbps), the retransmit rate limiter queues retransmissions until they can be sent within the rate limit. `naks_rx_delay_ignored` (above) will generally also rise if this count is high.

#### 7.65.1.13 **lbtm\_ulong\_t lbtm\_src\_transport\_stats\_lbtrm\_t\_stct::rx\_bytes\_sent**

Number of LBT-RM transport total bytes retransmitted by this source transport (triggered under the same circumstances as `rxs_sent`, above). In a normal, light-loss scenario, most NAKs induce a retransmission. When losses becomes heavy and/or

many receiver transports begin losing the same LBT-RM datagrams, NCF-related no-retransmit counts (naks\_ignored, naks\_shed and naks\_rx\_delay\_ignored) may begin to inflate, and retransmissions (rxs\_sent/rx\_bytes\_sent counts) may become significantly lower than NAKS received (naks\_rcved).

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.66 `lbtu_t_stct` Struct Reference

Structure that holds datagram statistics for source LBT-RU transports.

```
#include <lbtu.h>
```

### Data Fields

- `lbtu_t_stct::msgs_sent`
- `lbtu_t_stct::bytes_sent`
- `lbtu_t_stct::nak_pckts_rcved`
- `lbtu_t_stct::naks_rcved`
- `lbtu_t_stct::naks_ignored`
- `lbtu_t_stct::naks_shed`
- `lbtu_t_stct::naks_rx_delay_ignored`
- `lbtu_t_stct::rxs_sent`
- `lbtu_t_stct::num_clients`
- `lbtu_t_stct::rx_bytes_sent`

#### 7.66.1 Field Documentation

##### 7.66.1.1 `lbtu_t_stct::msgs_sent`

Number of LBT-RU datagrams sent. Depending on batching settings, a single LBT-RU datagram may contain one or more messages, or a fragment of a larger message. With LBT-RU, larger messages are split into fragment sizes limited by configuration option `transport_lbtu_datagram_max_size` (default 8KB).

##### 7.66.1.2 `lbtu_t_stct::bytes_sent`

Number of LBT-RU datagram bytes sent, i.e., the total of lengths of all LBT-RU packets including UM header information.

##### 7.66.1.3 `lbtu_t_stct::nak_pckts_rcved`

Number of NAK packets received by this source transport. UM batches NAKs into NAK packets to save network bandwidth. This should always be less than or equal to `naks_rcved` (below).

#### 7.66.1.4 `lbt_ulong_t lbt_src_transport_stats_lbtru_t_stct::naks_rcved`

Number of individual NAKs received by the source transport. When a source transport receives a NAK from a receiver transport, it may respond by re-transmitting the requested LBT-RU datagram, or it may send an NCF. The NAKing receiver transport responds to the NCF by waiting (timeout set by `transport_lbtru_nak_suppress_interval`, default 1000 ms), then re-sending the NAK.

#### 7.66.1.5 `lbt_ulong_t lbt_src_transport_stats_lbtru_t_stct::naks_ignored`

Number of NAKs this source transport ignored and sent an NCF with reason code "ignored". A source transport ignores a NAK for a datagram it has already recently retransmitted. How "recently" is determined by the configuration option `source_transport_lbtru_ignore_interval` (default 500ms). If this count is high, a receiver transport may be having trouble receiving retransmissions, or the ignore interval may be set too long.

#### 7.66.1.6 `lbt_ulong_t lbt_src_transport_stats_lbtru_t_stct::naks_shed`

Number of NAKs this source transport has shed by sending an NCF with reason code "shed". When a source transport's retransmit rate limiter and retransmit queue are both at maximum, it responds to a NAK by sending an "NCF shed", and does not retransmit. The receiver transport should wait, then send another NAK. If this count is high, one or more crybaby receiver transports may be clogging the source transport's retransmit queue.

#### 7.66.1.7 `lbt_ulong_t lbt_src_transport_stats_lbtru_t_stct::naks_rx_delay_ignored`

Number of NAKs this source transport has not yet processed because doing so would exceed its retransmit rate limit (set by configuration option `transport_lbtru_retransmit_rate_limit`, default 5Mbps). For each of these NAKs, the source transport immediately sends an NFC rx\_delay, then queues the retransmission for a later send within the rate limit. If this count is high, one or more crybaby receiver transports may be clogging the source transport's retransmit queue.

#### 7.66.1.8 `lbt_ulong_t lbt_src_transport_stats_lbtru_t_stct::rxs_sent`

Number of LBT-RU datagrams retransmitted by this source transport (incremented under the same circumstances as `rx_bytes_sent`, below). In a normal, light-loss scenario, most NAKs induce a retransmission. When losses becomes heavy and/or many receiver transports begin losing the same LBT-RU datagrams, NCF-related no-retransmit counts (`naks_ignored`, `naks_shed` and `naks_rx_delay_ignored`) may begin to inflate,

and retransmissions (rxs\_sent/rx\_bytes\_sent counts) may become significantly lower than NAKS received (naks\_rcved).

#### 7.66.1.9 `lbt_src_transport_stats_lbtru_t_stct::num_clients`

Number of receiver transports that are currently connected to this source transport.

#### 7.66.1.10 `lbt_src_transport_stats_lbtru_t_stct::rx_bytes_sent`

Number of LBT-RU transport total bytes retransmitted by this source transport (triggered under the same circumstances as rxs\_sent, above). In a normal, light-loss scenario, most NAKs induce a retransmission. When losses becomes heavy and/or many receiver transports begin losing the same LBT-RU datagrams, NCF-related no-retransmit counts (naks\_ignored, naks\_shed and naks\_rx\_delay\_ignored) may begin to inflate, and retransmissions (rxs\_sent/rx\_bytes\_sent counts) may become significantly lower than NAKS received (naks\_rcved).

The documentation for this struct was generated from the following file:

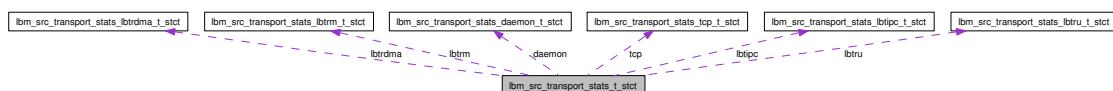
- [lbt.h](#)

## 7.67 `lbtm` Struct Reference

Structure that holds statistics for source transports.

```
#include <lbtm.h>
```

Collaboration diagram for `lbtm`:



## Data Fields

- int `type`
- char `source` [LBM\_MSG\_MAX\_SOURCE\_LEN]
- union {
 `lbtrdma`
`lbtrm`
`daemon`
`tcp`
`lbtru`
`lbtipc`
`lbtrdma`
} `transport`
- char `_fill` [LBM\_EXTERNAL\_STRUCT\_FILL\_SIZE]

### 7.67.1 Detailed Description

This structure holds statistics for all source transports. The structure is filled in when statistics for source transports are requested.

### 7.67.2 Field Documentation

#### 7.67.2.1 int `lbtm::type`

Type of transport (LBM\_TRANSPORT\_STAT\_TCP, LBM\_TRANSPORT\_STAT\_LBTRM, etc.).

**7.67.2.2 `char lbm_src_transport_stats_t_stct::source[LBM_MSG_MAX_SOURCE_LEN]`**

Source string of transport session, the format of which depends on the transport type.  
For string formats and examples, see [lbm\\_transport\\_source\\_info\\_t\\_stct](#).

**7.67.2.3 `lbm_src_transport_stats_tcp_t lbm_src_transport_stats_t_stct::tcp`**

The statistics for source TCP transports.

**7.67.2.4 `lbm_src_transport_stats_lbtrm_t lbm_src_transport_stats_t_stct::lbtrm`**

The statistics for source LBT-RM transports.

**7.67.2.5 `lbm_src_transport_stats_daemon_t lbm_src_transport_stats_t_stct::daemon`**

These statistics have been deprecated.

**7.67.2.6 `lbm_src_transport_stats_lbtru_t lbm_src_transport_stats_t_stct::lbtru`**

The statistics for source LBT-RU transports.

**7.67.2.7 `lbm_src_transport_stats_lbtrpc_t lbm_src_transport_stats_t_stct::lbtrpc`**

The statistics for source LBT-IPC transports.

**7.67.2.8 `lbm_src_transport_stats_lbtrdma_t lbm_src_transport_stats_t_stct::lbtrdma`**

The statistics for source LBT-RDMA transports.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.68 `lbt_src_transport_stats_tcp_t_stct` Struct Reference

Structure that holds datagram statistics for source TCP transports.

```
#include <lbt.h>
```

### Data Fields

- `lbt_ulong_t num_clients`
- `lbt_ulong_t bytes_buffered`

#### 7.68.1 Field Documentation

##### 7.68.1.1 `lbt_ulong_t lbt_src_transport_stats_tcp_t_stct::num_clients`

Number of TCP receiver clients currently connected over this transport.

##### 7.68.1.2 `lbt_ulong_t lbt_src_transport_stats_tcp_t_stct::bytes_buffered`

Number of bytes currently in UM's TCP buffer, i.e., a snapshot. This count is affected by the number of receivers, and configuration options `transport_tcp_multiple_receivers_behavior` and `transport_session_maximum_buffer`.

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.69 lbm\_str\_hash\_func\_ex\_t\_stct Struct Reference

Structure that holds the hash function callback information.

```
#include <lbm.h>
```

### Data Fields

- [lbtm\\_str\\_hash\\_function\\_cb\\_ex](#) hashfunc
- void \* clientd

#### 7.69.1 Detailed Description

A structure used with options to set/get a specific hash function information.

#### 7.69.2 Field Documentation

##### 7.69.2.1 lbm\_str\_hash\_function\_cb\_ex lbm\_str\_hash\_func\_ex\_t\_stct::hashfunc

Function pointer for hash function

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.70 lbtm\_timeval\_t\_stct Struct Reference

Structure that holds seconds and microseconds since midnight, Jan 1, 1970 UTC.

```
#include <lbtm.h>
```

### Data Fields

- lbm\_ulong\_t **tv\_sec**
- lbm\_ulong\_t **tv\_usec**

#### 7.70.1 Detailed Description

A structure included in UM messages to indicate when the message was received by UM. A message timestamp using this can be up to 500 milliseconds prior to actual receipt time, and hence, is not suitable when accurate message-arrival-time measurements are needed.

The documentation for this struct was generated from the following file:

- [lbtm.h](#)

## 7.71 lbm\_transport\_source\_info\_t\_stct Struct Reference

Structure that holds formatted and parsed transport source strings.

```
#include <lbm.h>
```

### Data Fields

- int `type`
- lbm\_uint32\_t `src_ip`
- lbm\_ushort\_t `src_port`
- lbm\_ushort\_t `dest_port`
- lbm\_uint32\_t `mc_group`
- lbm\_uint32\_t `transport_id`
- lbm\_uint32\_t `session_id`
- lbm\_uint32\_t `topic_idx`
- char `_fill` [LBM\_EXTERNAL\_STRUCT\_FILL\_SIZE]

#### 7.71.1 Detailed Description

This structure holds the fields used to format and/or parse transport source strings. The format of these strings depends mainly on the transport type, as shown below.

- TCP:src\_ip:src\_port[topic\_idx]  
example: TCP:192.168.0.4:45789[1539853954]
- LBTRM:src\_ip:src\_port:session\_id:mc\_group:dest\_port[topic\_idx]  
example: LBTRM:10.29.3.88:14390:e0679abb:231.13.13.13:14400[1539853954]
- LBT-RU:src\_ip:src\_port:session\_id[topic\_idx] (session\_id optional, per configuration option `transport_lbtru_use_session_id`)  
example: LBT-RU:192.168.3.189:34678[1539853954]
- LBT-IPC:session\_id:transport\_id[topic\_idx]  
example: LBT-IPC:6481f8d4:20000[1539853954]
- LBT-RDMA:src\_ip:src\_port:session\_id[topic\_idx]  
example: LBT-RDMA:192.168.3.189:34678:6471e9c4[1539853954]

Please note that the topic index field (topic\_idx) may or may not be present depending on your version of UM and/or the setting for configuration option source\_includes\_topic\_index.

**See also:**

[lbt\\_transport\\_source\\_format](#) [lbt\\_transport\\_source\\_parse](#)

## 7.71.2 Field Documentation

### 7.71.2.1 `int lbt_transport_source_info_t_stct::type`

Type of transport. See LBM\_TRANSPORT\_TYPE\_\*.

### 7.71.2.2 `lbt_uint32_t lbt_transport_source_info_t_stct::src_ip`

Source IP address. Applicable only to LBT-RM, LBT-RU, TCP, and LBT-RDMA. Stored in network order.

### 7.71.2.3 `lbt_ushort_t lbt_transport_source_info_t_stct::src_port`

Source port. Applicable only to LBT-RM, LBT-RU, TCP, and LBT-RDMA. Stored in host order.

### 7.71.2.4 `lbt_ushort_t lbt_transport_source_info_t_stct::dest_port`

Destination port. Applicable only to LBT-RM. Stored in host order.

### 7.71.2.5 `lbt_uint32_t lbt_transport_source_info_t_stct::mc_group`

Multicast group. Applicable only to LBT-RM. Stored in network order.

### 7.71.2.6 `lbt_uint32_t lbt_transport_source_info_t_stct::transport_id`

Transport ID. Applicable only to LBT-IPC. Stored in host order.

### 7.71.2.7 `lbt_uint32_t lbt_transport_source_info_t_stct::session_id`

Session ID. Applicable only to LBT-RM, LBT-RU, and LBT-IPC. Stored in host order.

**7.71.2.8 `lbtm_uint32_t lbtm_transport_source_info_t_stct::topic_idx`**

Topic index. Applicable to all transports. Stored in host order.

The documentation for this struct was generated from the following file:

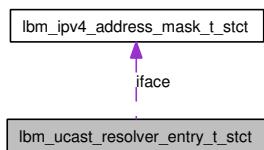
- [lbtm.h](#)

## 7.72 `lbtm_unicast_resolver_entry_t_stct` Struct Reference

Structure that holds information for a unicast resolver daemon for configuration purposes.

```
#include <lbtm.h>
```

Collaboration diagram for `lbtm_unicast_resolver_entry_t_stct`:



### Data Fields

- `lbtm_ipv4_address_mask_t_stct iface`
- `lbtm_uint_t resolver_ip`
- `lbtm_ushort_t source_port`
- `lbtm_ushort_t destination_port`

#### 7.72.1 Detailed Description

A structure used with options to get/set information about unicast resolver daemons.

#### 7.72.2 Field Documentation

##### 7.72.2.1 `lbtm_ipv4_address_mask_t lbtm_unicast_resolver_entry_t_stct::iface`

Interface to be used

##### 7.72.2.2 `lbtm_uint_t lbtm_unicast_resolver_entry_t_stct::resolver_ip`

The IP address of the unicast resolver daemon (network order)

##### 7.72.2.3 `lbtm_ushort_t lbtm_unicast_resolver_entry_t_stct::source_port`

The source port. Use 0 to indicate that the port should come from the [resolver\_unicast\_port\_low, resolver\_unicast\_port\_high] range. (network order)

**7.72.2.4 `lbt_ushort_t lbt_udcast_resolver_entry_t_stct::destination_port`**

The port configured on the unicast resolver daemon. (network order)

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.73 **lbtm\_ume\_ctx\_rcv\_ctx\_notification\_func\_t\_stct** Struct Reference

Structure that holds the application callback for receiving context status notifications for source context.

```
#include <lbtm.h>
```

### Data Fields

- `lbtm_ume_ctx_rcv_ctx_notification_create_function_cb` **create\_func**
- `lbtm_ume_ctx_rcv_ctx_notification_delete_function_cb` **delete\_func**
- `void * clientd`

#### 7.73.1 Detailed Description

A Structure used with options to set/get a specific callback function

The documentation for this struct was generated from the following file:

- [lbtm.h](#)

## 7.74 `lbt_ume_rcv_recovery_info_ex_func_info_t_stct` Struct Reference

Structure that holds information for UMP receiver recovery sequence number info application callbacks.

```
#include <lbt.h>
```

### Data Fields

- int `flags`
- lbt\_uint\_t `low_sequence_number`
- lbt\_uint\_t `low_rxreq_max_sequence_number`
- lbt\_uint\_t `high_sequence_number`
- void \* `source_clientd`
- char `source` [LBM\_MSG\_MAX\_SOURCE\_LEN]

### 7.74.1 Detailed Description

A structure used with UMP receiver recovery sequence number information callbacks to pass in information as well as return low sequence number information.

See also:

[lbt\\_ume\\_rcv\\_recovery\\_info\\_ex\\_func\\_t](#)

### 7.74.2 Field Documentation

#### 7.74.2.1 int `lbt_ume_rcv_recovery_info_ex_func_info_t_stct::flags`

Flags that indicate optional portions that are included

#### 7.74.2.2 lbt\_uint\_t `lbt_ume_rcv_recovery_info_ex_func_info_t_stct::low_sequence_number`

Lowest sequence number that the receiver will attempt to recover. May be altered to instruct UMP to recover differently

**7.74.2.3 `lbm_uint_t lbm_ume_rcv_recovery_info_ex_func_info_t_stct::low_rxreq_max_sequence_number`**

Lowest sequence number that the receiver will attempt to recover (with retransmit request maximum taken into account)

**7.74.2.4 `lbm_uint_t lbm_ume_rcv_recovery_info_ex_func_info_t_stct::high_sequence_number`**

Highest sequence number that the receiver has seen from the source. May not be actual highest sent by source

**7.74.2.5 `void* lbm_ume_rcv_recovery_info_ex_func_info_t_stct::source_clientd`**

The per-source clientd value for the source set by the `lbt_rcv_src_notification_create_function_cb` callback

**7.74.2.6 `char lbm_ume_rcv_recovery_info_ex_func_info_t_stct::source[LBM_MSG_MAX_SOURCE_LEN]`**

The source

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.75 `lbt_ume_rcv_recovery_info_ex_func_t_stct` Struct Reference

Structure that holds the application callback for recovery sequence number information, extended form.

```
#include <lbt.h>
```

### Data Fields

- `lbt_ume_rcv_recovery_info_ex_function_cb func`
- `void * clientd`

#### 7.75.1 Detailed Description

A struct used with options to set/get a specific callback function

The documentation for this struct was generated from the following file:

- `lbt.h`

## 7.76 `lbt_ume_rcv_regid_ex_func_info_t_stct` Struct Reference

Structure that holds information for UMP receiver registration ID application callbacks.

```
#include <lbt.h>
```

### Data Fields

- int `flags`
- `lbt_uint_t` `src_registration_id`
- `lbt_ushort_t` `store_index`
- `void *` `source_clientd`
- `char` `source` [`LBT_MSG_MAX_SOURCE_LEN`]
- `char` `store` [`LBT_UME_MAX_STORE_STRLEN`]

#### 7.76.1 Detailed Description

A structure used with UMP receiver registration ID callbacks to pass in information.

See also:

[lbt\\_ume\\_rcv\\_regid\\_func\\_t](#)

#### 7.76.2 Field Documentation

##### 7.76.2.1 `int lbt_ume_rcv_regid_ex_func_info_t_stct::flags`

Flags that indicate which optional portions are included

##### 7.76.2.2 `lbt_uint_t lbt_ume_rcv_regid_ex_func_info_t_stct::src_registration_id`

The registration ID for the source

##### 7.76.2.3 `lbt_ushort_t lbt_ume_rcv_regid_ex_func_info_t_stct::store_index`

The store index of the store involved

**7.76.2.4 void\* lbm\_ume\_rcv\_regid\_ex\_func\_info\_t\_stct::source\_clientd**

The per-source clientd value for the source set by the lbm\_rcv\_src\_notification\_create\_function\_cb callback

**7.76.2.5 char lbm\_ume\_rcv\_regid\_ex\_func\_info\_t\_stct::source[LBM\_MSG\_-MAX\_SOURCE\_LEN]**

The source for the registration ID

**7.76.2.6 char lbm\_ume\_rcv\_regid\_ex\_func\_info\_t\_stct::store[LBM\_UME\_-MAX\_STORE\_STRLEN]**

The store involved

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.77 `lbt_ume_rcv_regid_ex_func_t_stct` Struct Reference

Structure that holds the application callback for registration ID setting, extended form.

```
#include <lbt.h>
```

### Data Fields

- `lbt_ume_rcv_regid_ex_function_cb func`
- `void * clientd`

#### 7.77.1 Detailed Description

A structure used with options to set/get a specific callback function

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.78 `lbt_ume_rcv_regid_func_t_stct` Struct Reference

Structure that holds the application callback for registration ID setting.

```
#include <lbt.h>
```

### Data Fields

- `lbt_ume_rcv_regid_function_cb func`
- `void * clientd`

#### 7.78.1 Detailed Description

A structure used with options to set/get a specific callback function

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.79 **lbt\_ume\_src\_force\_reclaim\_func\_t\_stct** Struct Reference

Structure that holds the application callback for forced reclamation notifications.

```
#include <lbt.h>
```

### Data Fields

- **lbt\_ume\_src\_force\_reclaim\_function\_cb func**
- **void \* clientd**

#### 7.79.1 Detailed Description

A structure used with options to set/get a specific callback function

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.80 `lbt_ume_store_entry_t_stct` Struct Reference

Structure that holds information for a UMP store for configuration purposes.

```
#include <lbt.h>
```

### Data Fields

- `lbt_uint_t ip_address`
- `lbt_ushort_t tcp_port`
- `lbt_ushort_t group_index`
- `lbt_uint_t registration_id`

#### 7.80.1 Detailed Description

A structure used with options to get/set information for a UMP store

#### 7.80.2 Field Documentation

##### 7.80.2.1 `lbt_uint_t lbt_ume_store_entry_t_stct::ip_address`

The IP address of the UMP store (network order)

##### 7.80.2.2 `lbt_ushort_t lbt_ume_store_entry_t_stct::tcp_port`

The TCP port of the store (network order)

##### 7.80.2.3 `lbt_ushort_t lbt_ume_store_entry_t_stct::group_index`

The index of the group this UMP store belongs to

##### 7.80.2.4 `lbt_uint_t lbt_ume_store_entry_t_stct::registration_id`

The registration ID that should be used with this UMP store for the source

The documentation for this struct was generated from the following file:

- `lbt.h`

## 7.81 `lbt_ume_store_group_entry_t_stct` Struct Reference

Structure that holds information for a UMP store group for configuration purposes.

```
#include <lbt.h>
```

### Data Fields

- `lbt_ushort_t index`
- `lbt_ushort_t group_size`

#### 7.81.1 Detailed Description

A structure used with options to get/set information for a UMP store group

#### 7.81.2 Field Documentation

##### 7.81.2.1 `lbt_ushort_t lbt_ume_store_group_entry_t_stct::index`

Index of the group. Used in the individual store entries to indicate the store is in this group

##### 7.81.2.2 `lbt_ushort_t lbt_ume_store_group_entry_t_stct::group_size`

The size of the group

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.82 **lbt\_ume\_store\_name\_entry\_t\_stct** Struct Reference

Structure that holds information for a UMP store by name for configuration purposes.

```
#include <lbt.h>
```

### Data Fields

- char [name](#) [LBM\_MAX\_CONTEXT\_NAME\_LEN+1]
- lbt\_ushort\_t [group\\_index](#)
- lbt\_uint\_t [registration\\_id](#)

#### 7.82.1 Detailed Description

A structure used with options to get/set information for a UMP store

#### 7.82.2 Field Documentation

##### 7.82.2.1 **char lbt\_ume\_store\_name\_entry\_t\_stct::name[LBM\_MAX\_CONTEXT\_NAME\_LEN+1]**

The store context name Restricted to alphanumeric characters, hyphens, and underscores

##### 7.82.2.2 **lbt\_ushort\_t lbt\_ume\_store\_name\_entry\_t\_stct::group\_index**

The index of the group this UMP store belongs to

##### 7.82.2.3 **lbt\_uint\_t lbt\_ume\_store\_name\_entry\_t\_stct::registration\_id**

The registration ID that should be used with this UMP store for the source

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.83 `lbt_umm_info_t_stct` Struct Reference

Structure for specifying UMM daemon connection options.

```
#include <lbt.h>
```

### Data Fields

- char `username` [LBM\_UMM\_USER\_NAME\_LENGTH\_MAX]
- char `password` [LBM\_UMM\_PASSWORD\_LENGTH\_MAX]
- char `appname` [LBM\_UMM\_APP\_NAME\_LENGTH\_MAX]
- char `servers` [LBM\_UMM\_NUM\_SERVERS\_MAX][LBM\_UMM\_SERVER\_LENGTH\_MAX]
- `lbt_uint32_t flags`
- char \* `cert_file`
- char \* `cert_file_password`

#### 7.83.1 Field Documentation

##### 7.83.1.1 char `lbt_umm_info_t_stct::username`[LBM\_UMM\_USER\_NAME\_LENGTH\_MAX]

The UMM user name.

##### 7.83.1.2 char `lbt_umm_info_t_stct::password`[LBM\_UMM\_PASSWORD\_LENGTH\_MAX]

The UMM user password.

##### 7.83.1.3 char `lbt_umm_info_t_stct::appname`[LBM\_UMM\_APP\_NAME\_LENGTH\_MAX]

The UMM application name.

##### 7.83.1.4 char `lbt_umm_info_t_stct::servers`[LBM\_UMM\_NUM\_SERVERS\_MAX][LBM\_UMM\_SERVER\_LENGTH\_MAX]

The list of UMM daemons in ip:port string format. Connections are tried in a round robin fashion, starting with index 0. Only the first contiguous set of non-blank entries are considered.

**7.83.1.5 `lbt_uint32_t lbt_umm_info_t_stct::flags`**

Flags, currently used to enable SSL.

**7.83.1.6 `char* lbt_umm_info_t_stct::cert_file`**

Path to a pem-encoded certificate file. If non-NULL, SSL is enabled and is used to validate the the UMM daemon certificate.

**7.83.1.7 `char* lbt_umm_info_t_stct::cert_file_password`**

Certificate file password. Required only if certificate file is password-protected.

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.84 `lbt_umq_index_info_t_stct` Struct Reference

Structure that holds information used for sending and receiving messages with UMQ indices.

```
#include <lbt.h>
```

### Data Fields

- int `flags`
- int `reserved`
- char `index` [LBM\_UMQ\_MAX\_INDEX\_LEN]
- size\_t `index_len`

#### 7.84.1 Detailed Description

A structure used with UM sources and receivers to associate UMQ Indices with messages.

#### 7.84.2 Field Documentation

##### 7.84.2.1 int `lbt_umq_index_info_t_stct::flags`

Flags that indicate optional portions are included

##### 7.84.2.2 char `lbt_umq_index_info_t_stct::index[LBM_UMQ_MAX_INDEX_LEN]`

The index

##### 7.84.2.3 size\_t `lbt_umq_index_info_t_stct::index_len`

The length of the index in bytes

The documentation for this struct was generated from the following file:

- `lbt.h`

## 7.85 `lbt_umq_msg_total_lifetime_info_t_stct` Struct Reference

Structure that holds UMQ message total lifetime information.

```
#include <lbt.h>
```

### Data Fields

- int `flags`
- lbt\_ulong\_t `umq_msg_total_lifetime`

#### 7.85.1 Detailed Description

A structure used with UMQ sources to specify a message's total lifetime.

#### 7.85.2 Field Documentation

##### 7.85.2.1 int `lbt_umq_msg_total_lifetime_info_t_stct::flags`

Flags that indicate which optional portions are included

##### 7.85.2.2 lbt\_ulong\_t `lbt_umq_msg_total_lifetime_info_t_stct::umq_msg_total_lifetime`

The message's total lifetime, in milliseconds

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.86 `lbt_umq_mgid_t_stct` Struct Reference

Structure that holds information for UMQ messages that allows the message to be identified uniquely.

```
#include <lbt.h>
```

### Data Fields

- `lbt_umq_regid_t regid`
- `lbt_uint64_t stamp`

#### 7.86.1 Detailed Description

See also:

`lbt_umq_regid_t` A structure used with UMQ messages to identify a message uniquely.

#### 7.86.2 Field Documentation

##### 7.86.2.1 `lbt_umq_regid_t lbt_umq_mgid_t_stct::regid`

The Registration ID of the original source context of the message

##### 7.86.2.2 `lbt_uint64_t lbt_umq_mgid_t_stct::stamp`

The stamp of the message that indicates the individual message from the given source context

The documentation for this struct was generated from the following file:

- `lbt.h`

## 7.87 `lbt_umq_queue_entry_t_stct` Struct Reference

Structure that holds information for a UMQ queue registration ID for configuration purposes.

```
#include <lbt.h>
```

### Data Fields

- char `name` [LBM\_UMQ\_MAX\_QUEUE\_STRLEN]
- `lbt_umq_regid_t regid`

#### 7.87.1 Detailed Description

A struct used with options to get/set Registration ID information for UMQ queues

#### 7.87.2 Field Documentation

##### 7.87.2.1 char `lbt_umq_queue_entry_t_stct::name`[LBM\_UMQ\_MAX\_QUEUE\_STRLEN]

Name of the UMQ queue

##### 7.87.2.2 `lbt_umq_regid_t lbt_umq_queue_entry_t_stct::regid`

Registration ID to use with the given UMQ queue

The documentation for this struct was generated from the following file:

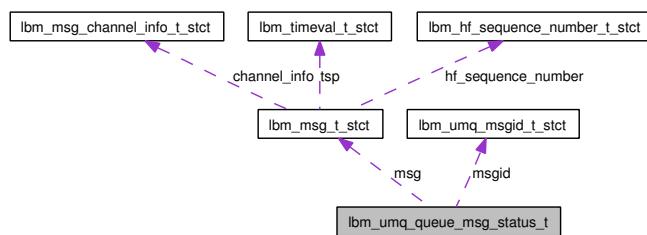
- `lbt.h`

## 7.88 `lbt_umq_queue_msg_status_t` Struct Reference

Struct containing extended asynchronous operation status information about a single UMQ message.

```
#include <lbt.h>
```

Collaboration diagram for `lbt_umq_queue_msg_status_t`:



### Data Fields

- `lbt_umqmsgid_t msgid`
- `lbt_msg_t * msg`
- `void * clientd`
- `int status`
- `int flags`

#### 7.88.1 Field Documentation

##### 7.88.1.1 `lbt_umqmsgid_t lbt_umq_queue_msg_status_t::msgid`

UMQ message ID of this message.

##### 7.88.1.2 `lbt_msg_t* lbt_umq_queue_msg_status_t::msg`

Actual message, if appropriate and available (NULL otherwise)

##### 7.88.1.3 `void* lbt_umq_queue_msg_status_t::clientd`

User client data pointer.

**7.88.1.4 int `lbt_umq_queue_msg_status_t::status`**

Status code for this message (retrieval in progress, consumed, etc.)

**7.88.1.5 int `lbt_umq_queue_msg_status_t::flags`**

Reserved flags field.

The documentation for this struct was generated from the following file:

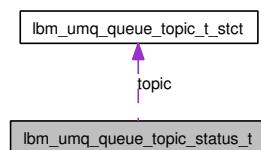
- [lbt.h](#)

## 7.89 `lbt_umq_queue_topic_status_t` Struct Reference

Struct containing extended asynchronous operation status information about a single UMQ topic.

```
#include <lbt.h>
```

Collaboration diagram for `lbt_umq_queue_topic_status_t`:



### Data Fields

- `lbt_umq_queue_topic_t * topic`
- int `status`
- int `flags`

#### 7.89.1 Field Documentation

##### 7.89.1.1 `lbt_umq_queue_topic_t* lbt_umq_queue_topic_status_t::topic`

UMQ topic info.

##### 7.89.1.2 `int lbt_umq_queue_topic_status_t::status`

Status code for this topic (reserved for future use; will currently always be set to 0).

##### 7.89.1.3 `int lbt_umq_queue_topic_status_t::flags`

Reserved flags field.

The documentation for this struct was generated from the following file:

- `lbt.h`

## 7.90 `lbt_umq_queue_topic_t_stct` Struct Reference

Structure that holds queue topic information and can be used as a handle to a queue topic.

```
#include <lbt.h>
```

### Data Fields

- char `topic_name` [LBM\_UMQ\_MAX\_TOPIC\_STRLEN]
- `lbt_umq_queue_application_set_t *` `appsets`
- int `num_appsets`
- void \* `reserved`

#### 7.90.1 Field Documentation

##### 7.90.1.1 char `lbt_umq_queue_topic_t_stct::topic_name`[LBM\_UMQ\_MAX\_TOPIC\_STRLEN]

The topic name string.

##### 7.90.1.2 `lbt_umq_queue_application_set_t* lbt_umq_queue_topic_t_stct::appsets`

Array of application sets within this topic.

##### 7.90.1.3 int `lbt_umq_queue_topic_t_stct::num_appsets`

Length of the application sets array.

##### 7.90.1.4 void\* `lbt_umq_queue_topic_t_stct::reserved`

Reserved field; do not touch.

The documentation for this struct was generated from the following file:

- `lbt.h`

## 7.91 `lqm_umq_ulb_application_set_attr_t_stct` Struct Reference

Structure that holds information for a UMQ ULB sources application set attributes.

```
#include <lqm.h>
```

### Data Fields

- `lqm_ushort_t index`
- union {
  - int `d`
  - `lqm_ulong_t lu`}
- `value`

#### 7.91.1 Detailed Description

A struct used with options to get/set UMQ ULB application set attributes

#### 7.91.2 Field Documentation

##### 7.91.2.1 `lqm_ushort_t lqm_umq_ulb_application_set_attr_t_stct::index`

Index of the Application Set

##### 7.91.2.2 `int lqm_umq_ulb_application_set_attr_t_stct::d`

The integer value of the attribute

##### 7.91.2.3 `lqm_ulong_t lqm_umq_ulb_application_set_attr_t_stct::lu`

The unsigned long int value of the attribute

##### 7.91.2.4 `union { ... } lqm_umq_ulb_application_set_attr_t_stct::value`

The value of the attribute

The documentation for this struct was generated from the following file:

- [lqm.h](#)

## 7.92 **lmb\_umq\_ulb\_receiver\_type\_attr\_t\_stct** Struct Reference

Structure that holds information for a UMQ ULB sources receiver type attributes.

```
#include <lmb.h>
```

### Data Fields

- `lmb_ulong_t id`
- union {  
    int `d`  
    lmb\_ulong\_t `lu`  
} `value`

#### 7.92.1 Detailed Description

A struct used with options to get/set UMQ ULB receiver type attributes

#### 7.92.2 Field Documentation

##### 7.92.2.1 `lmb_ulong_t lmb_umq_ulb_receiver_type_attr_t_stct::id`

Receiver Type ID

##### 7.92.2.2 `int lmb_umq_ulb_receiver_type_attr_t_stct::d`

The integer value of the attribute

##### 7.92.2.3 `lmb_ulong_t lmb_umq_ulb_receiver_type_attr_t_stct::lu`

The unsigned long int value of the attribute

##### 7.92.2.4 `union { ... } lmb_umq_ulb_receiver_type_attr_t_stct::value`

The value of the attribute

The documentation for this struct was generated from the following file:

- `lmb.h`

## 7.93 `lbt_umq_ulb_receiver_type_entry_t_stct` Struct Reference

Structure that holds information for a UMQ ULB sources receiver type associations with application sets.

```
#include <lbt.h>
```

### Data Fields

- `lbt_ulong_t id`
- `lbt_ushort_t application_set_index`

#### 7.93.1 Detailed Description

A struct used with options to get/set UMQ ULB Receiver Type entries

#### 7.93.2 Field Documentation

##### 7.93.2.1 `lbt_ulong_t lbt_umq_ulb_receiver_type_entry_t_stct::id`

Receiver Type ID

##### 7.93.2.2 `lbt_ushort_t lbt_umq_ulb_receiver_type_entry_t_stct::application_set_index`

Index of the Application Set the receiver is in

The documentation for this struct was generated from the following file:

- `lbt.h`

## 7.94 lbm\_wildcard\_rcv\_compare\_func\_t\_stct Struct Reference

Structure that holds the application callback pattern type information for wildcard receivers.

```
#include <lbm.h>
```

### Data Fields

- [lbm\\_wildcard\\_rcv\\_compare\\_function\\_cb compfunc](#)
- [void \\* clientd](#)

#### 7.94.1 Detailed Description

A structure used with options to set/get a specific application callback pattern type.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.95 `lbt_wildcard_rcv_create_func_t_stct` Struct Reference

Structure that holds the receiver creation callback information for wildcard receivers.

```
#include <lbt.h>
```

### Data Fields

- `lbt_wildcard_rcv_create_function_cb` **createfunc**
- `void *` **clientd**

#### 7.95.1 Detailed Description

A structure used with options to set/get a specific wildcard topic receiver creation callback type.

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.96 lbm\_wildcard\_rcv\_delete\_func\_t\_stct Struct Reference

Structure that holds the receiver deletion callback information for wildcard receivers.

```
#include <lbm.h>
```

### Data Fields

- [lbt\\_wildcard\\_rcv\\_delete\\_function\\_cb deletefunc](#)
- void \* clientd

#### 7.96.1 Detailed Description

A structure used with options to set/get a specific wildcard topic receiver deletion callback type.

The documentation for this struct was generated from the following file:

- [lbt.h](#)

## 7.97 lbmmon\_attr\_block\_t\_stct Struct Reference

Statistics attribute block layout. Associated with each statistics message is a set of optional attributes. Any attributes present will immediately follow the packet header.

```
#include <lbmmon.h>
```

### Data Fields

- lbm\_ushort\_t [mEntryCount](#)
- lbm\_ushort\_t [mEntryLength](#)

#### 7.97.1 Field Documentation

##### 7.97.1.1 lbm\_ushort\_t lbmmon\_attr\_block\_t\_stct::mEntryCount

Number of attribute entries in network order.

##### 7.97.1.2 lbm\_ushort\_t lbmmon\_attr\_block\_t\_stct::mEntryLength

Total length of the attribute block in network order.

The documentation for this struct was generated from the following file:

- [lbmmon.h](#)

## 7.98 lbmmon\_attr\_entry\_t\_stct Struct Reference

Statistics attribute entry layout. Each attribute entry within the attributes block consists of an entry header, followed immediately by the attribute data.

```
#include <lbmmmon.h>
```

### Data Fields

- lbm\_ushort\_t [mType](#)
- lbm\_ushort\_t [mLength](#)

#### 7.98.1 Field Documentation

##### 7.98.1.1 lbm\_ushort\_t lbmmon\_attr\_entry\_t\_stct::mType

Attribute type in network order.

##### 7.98.1.2 lbm\_ushort\_t lbmmon\_attr\_entry\_t\_stct::mLength

Length of attribute data for this entry in network order.

The documentation for this struct was generated from the following file:

- [lbmmmon.h](#)

## 7.99 lbmon\_ctx\_statistics\_func\_t\_stct Struct Reference

A structure that holds the callback information for context statistics.

```
#include <lbmon.h>
```

### Data Fields

- [lbmon\\_ctx\\_statistics\\_cb cbfunc](#)

#### 7.99.1 Detailed Description

A structure used with receive controller options to get/set specific callback information.

#### 7.99.2 Field Documentation

##### 7.99.2.1 lbmon\_ctx\_statistics\_cb lbmon\_ctx\_statistics\_func\_t\_stct::cbfunc

Context statistics callback function.

The documentation for this struct was generated from the following file:

- [lbmon.h](#)

## 7.100 lbmmon\_evq\_statistics\_func\_t\_stct Struct Reference

A structure that holds the callback information for event queue statistics.

```
#include <lbmmon.h>
```

### Data Fields

- [lbmmon\\_evq\\_statistics\\_cb cbfunc](#)

#### 7.100.1 Detailed Description

A structure used with receive controller options to get/set specific callback information.

#### 7.100.2 Field Documentation

##### 7.100.2.1 lbmmon\_evq\_statistics\_cb lbmmon\_evq\_statistics\_func\_t\_stct::cbfunc

Event queue statistics callback function.

The documentation for this struct was generated from the following file:

- [lbmmon.h](#)

## 7.101 lbmon\_format\_func\_t\_stct Struct Reference

Format module function pointer container.

```
#include <lbmon.h>
```

### Data Fields

- `lbmon_format_init_t mInit`
- `lbmon_rcv_format_serialize_t mRcvSerialize`
- `lbmon_src_format_serialize_t mSrcSerialize`
- `lbmon_rcv_format_deserialize_t mRcvDeserialize`
- `lbmon_src_format_deserialize_t mSrcDeserialize`
- `lbmon_format_finish_t mFinish`
- `lbmon_format_errmsg_t mErrorMessage`
- `lbmon_evq_format_serialize_t mEvqSerialize`
- `lbmon_evq_format_deserialize_t mEvqDeserialize`
- `lbmon_ctx_format_serialize_t mCtxSerialize`
- `lbmon_ctx_format_deserialize_t mCtxDeserialize`

### 7.101.1 Field Documentation

#### 7.101.1.1 lbmon\_format\_init\_t lbmon\_format\_func\_t\_stct::mInit

Initialization function.

#### 7.101.1.2 lbmon\_rcv\_format\_serialize\_t lbmon\_format\_func\_t\_stct::mRcvSerialize

Function to serialize receiver statistics (lbm\_rcv\_transport\_stats\_t).

#### 7.101.1.3 lbmon\_src\_format\_serialize\_t lbmon\_format\_func\_t\_stct::mSrcSerialize

Function to serialize source statistics (lbm\_src\_transport\_stats\_t).

#### 7.101.1.4 lbmon\_rcv\_format\_deserialize\_t lbmon\_format\_func\_t\_stct::mRcvDeserialize

Function to deserialize receiver statistics (lbm\_rcv\_transport\_stats\_t).

**7.101.1.5 lbmmon\_src\_format\_deserialize\_t lbmmon\_format\_func\_t -  
stct::mSrcDeserialize**

Function to deserialize source statistics (lmb\_src\_transport\_stats\_t).

**7.101.1.6 lbmmon\_format\_finish\_t lbmmon\_format\_func\_t\_stct::mFinish**

Format-specific cleanup function.

**7.101.1.7 lbmmon\_format\_errmsg\_t lbmmon\_format\_func\_t -  
stct::mErrorMessage**

Function to return a message describing the last error encountered.

**7.101.1.8 lbmmon\_evq\_format\_serialize\_t lbmmon\_format\_func\_t -  
stct::mEvqSerialize**

Function to serialize event queue statistics (lmb\_event\_queue\_stats\_t).

**7.101.1.9 lbmmon\_evq\_format\_deserialize\_t lbmmon\_format\_func\_t -  
stct::mEvqDeserialize**

Function to deserialize event queue statistics (lmb\_event\_queue\_stats\_t).

**7.101.1.10 lbmmon\_ctx\_format\_serialize\_t lbmmon\_format\_func\_t -  
stct::mCtxSerialize**

Function to serialize context statistics (lmb\_context\_stats\_t).

**7.101.1.11 lbmmon\_ctx\_format\_deserialize\_t lbmmon\_format\_func\_t -  
stct::mCtxDeserialize**

Function to deserialize context statistics (lmb\_context\_stats\_t).

The documentation for this struct was generated from the following file:

- [lbmmon.h](#)

## 7.102 lbmmon\_packet\_hdr\_t\_stct Struct Reference

Statistics packet header layout.

```
#include <lbmmon.h>
```

### Data Fields

- `lbt_uint_t mSignature`
- `lbt_ushort_t mType`
- `lbt_ushort_t mAttributeBlockLength`
- `lbt_ushort_t mDataLength`
- `lbt_ushort_t mFiller`

### 7.102.1 Detailed Description

A statistics packet consists of four fixed-length and fixed-position fields, as documented below. It is followed by two variable-length fields. The option block is located at `packet + sizeof(lbmmon_packet_hdr_t)`, and is `mOptionBlockLength` (when properly interpreted) bytes in length (which may be zero). The statistics data block is located immediately following the option block.

### 7.102.2 Field Documentation

#### 7.102.2.1 lbt\_uint\_t lbmmon\_packet\_hdr\_t\_stct::mSignature

Packet signature in network order. Must be [LBMMON\\_PACKET\\_SIGNATURE](#).

#### 7.102.2.2 lbt\_ushort\_t lbmmon\_packet\_hdr\_t\_stct::mType

Type of statistics, in network order. See [LBMMON\\_PACKET\\_TYPE\\_\\*](#).

#### 7.102.2.3 lbt\_ushort\_t lbmmon\_packet\_hdr\_t\_stct::mAttributeBlockLength

Length of optional, variable-length attribute block in network order.

#### 7.102.2.4 lbt\_ushort\_t lbmmon\_packet\_hdr\_t\_stct::mDataLength

Length of variable-length statistics data in network order.

**7.102.2.5 `lbm_ushort_t lbmmon_packet_hdr_t::stct::mFiller`**

Filler to assure proper alignment of the structure.

The documentation for this struct was generated from the following file:

- [lbmmon.h](#)

## 7.103 lbmon\_rcv\_statistics\_func\_t\_stct Struct Reference

A structure that holds the callback information for receiver statistics.

```
#include <lbmon.h>
```

### Data Fields

- [lbmon\\_rcv\\_statistics\\_cb cbfunc](#)

#### 7.103.1 Detailed Description

A structure used with receive controller options to get/set specific callback information.

#### 7.103.2 Field Documentation

##### 7.103.2.1 lbmon\_rcv\_statistics\_cb lbmon\_rcv\_statistics\_func\_t\_stct::cbfunc

Receiver statistics callback function.

The documentation for this struct was generated from the following file:

- [lbmon.h](#)

## 7.104 lbmmmon\_src\_statistics\_func\_t\_stct Struct Reference

A structure that holds the callback information for source statistics.

```
#include <lbmmmon.h>
```

### Data Fields

- [lbmmmon\\_src\\_statistics\\_cb cbfunc](#)

#### 7.104.1 Detailed Description

A structure used with receive controller options to get/set specific callback information.

#### 7.104.2 Field Documentation

##### 7.104.2.1 lbmmmon\_src\_statistics\_cb lbmmmon\_src\_statistics\_func\_t\_stct::cbfunc

Source statistics callback function.

The documentation for this struct was generated from the following file:

- [lbmmmon.h](#)

## 7.105 lbmon\_transport\_func\_t\_stct Struct Reference

Transport module function pointer container.

```
#include <lbmon.h>
```

### Data Fields

- `lbmon_transport_initsrc_t mInitSource`
- `lbmon_transport_initrcv_t mInitReceiver`
- `lbmon_transport_send_t mSend`
- `lbmon_transport_receive_t mReceive`
- `lbmon_transport_finishsrc_t mFinishSource`
- `lbmon_transport_finishrcv_t mFinishReceiver`
- `lbmon_transport_errmsg_t mErrorMessage`

### 7.105.1 Field Documentation

#### 7.105.1.1 lbmon\_transport\_initsrc\_t lbmon\_transport\_func\_t\_stct::mInitSource

Initialize module as a statistics source.

#### 7.105.1.2 lbmon\_transport\_initrcv\_t lbmon\_transport\_func\_t\_stct::mInitReceiver

Initialize module as a statistics receiver.

#### 7.105.1.3 lbmon\_transport\_send\_t lbmon\_transport\_func\_t\_stct::mSend

Send a statistics packet.

#### 7.105.1.4 lbmon\_transport\_receive\_t lbmon\_transport\_func\_t\_stct::mReceive

Receive statistics data.

#### 7.105.1.5 lbmon\_transport\_finishsrc\_t lbmon\_transport\_func\_t\_stct::mFinishSource

Finish processing for a source transport.

**7.105.1.6 lbmmon\_transport\_finishrev\_t lbmmon\_transport\_func\_t -  
stct:::mFinishReceiver**

Finish processing for a receiver transport.

**7.105.1.7 lbmmon\_transport\_errmsg\_t lbmmon\_transport\_func\_t -  
stct:::mErrorMessage**

Function to return a message describing the last error encountered.

The documentation for this struct was generated from the following file:

- [lbmmon.h](#)

## 7.106 lbmpdm\_decimal\_t Struct Reference

Structure to hold a scaled decimal number. A scaled decimal number consists of a mantissa  $m$  and an exponent  $exp$ . It represents the value  $m \cdot 10^{exp}$ .

```
#include <lbmpdm.h>
```

### Data Fields

- int64\_t `mant`
- int8\_t `exp`

#### 7.106.1 Detailed Description

The mantissa is represented as a 64-bit signed integer. The exponent is represented as an 8-bit signed integer, and can range from -128 to 127.

#### 7.106.2 Field Documentation

##### 7.106.2.1 int64\_t lbmpdm\_decimal\_t::mant

Mantissa.

##### 7.106.2.2 int8\_t lbmpdm\_decimal\_t::exp

Exponent.

The documentation for this struct was generated from the following file:

- [lbmpdm.h](#)

## 7.107 lbmpdm\_field\_info\_attr\_stct\_t Struct Reference

Attribute struct to be passed along with the name when adding field info to a definition.

```
#include <lbmpdm.h>
```

### Data Fields

- int **flags**
- uint8\_t **req**
- uint32\_t **fixed\_str\_len**
- uint32\_t **num\_arr\_elem**
- char **fill** [256]

### 7.107.1 Field Documentation

#### 7.107.1.1 uint8\_t lbmpdm\_field\_info\_attr\_stct\_t::req

If the field is required or not.

#### 7.107.1.2 uint32\_t lbmpdm\_field\_info\_attr\_stct\_t::fixed\_str\_len

The number of characters for fixed string or fixed unicode types.

#### 7.107.1.3 uint32\_t lbmpdm\_field\_info\_attr\_stct\_t::num\_arr\_elem

The number of elements for fixed arrays.

The documentation for this struct was generated from the following file:

- [lbmpdm.h](#)

## 7.108 lbmpdm\_field\_value\_stct\_t Struct Reference

Field value struct that can be populated with a field value when passed to the lbmpdm\_-msg\_get\_field\_value\_stct function.

```
#include <lbmpdm.h>
```

### Data Fields

- `uint16_t field_type`
- `uint8_t is_array`
- `uint8_t is_fixed`
- `size_t len`
- `char * value`
- `uint32_t num_arr_elem`
- `size_t * len_arr`
- `char ** value_arr`
- `char fill [256]`

### 7.108.1 Field Documentation

#### 7.108.1.1 uint16\_t lbmpdm\_field\_value\_stct\_t::field\_type

The field type.

#### 7.108.1.2 uint8\_t lbmpdm\_field\_value\_stct\_t::is\_array

If the field is an array

#### 7.108.1.3 uint8\_t lbmpdm\_field\_value\_stct\_t::is\_fixed

If the field is a fixed length field.

#### 7.108.1.4 size\_t lbmpdm\_field\_value\_stct\_t::len

The length in bytes of the field for scalar fields.

#### 7.108.1.5 char\* lbmpdm\_field\_value\_stct\_t::value

A pointer to the field value for scalar fields.

**7.108.1.6 uint32\_t lbmpdm\_field\_value\_stct\_t::num\_arr\_elem**

The number of elements in the array for array fields.

**7.108.1.7 size\_t\* lbmpdm\_field\_value\_stct\_t::len\_arr**

An array of size\_t representing the length in bytes for each array element for array fields.

**7.108.1.8 char\*\* lbmpdm\_field\_value\_stct\_t::value\_arr**

An array of pointers to the field values for array fields.

The documentation for this struct was generated from the following file:

- [lbmpdm.h](#)

## 7.109 lbmpdm\_timestamp\_t Struct Reference

Structure to hold a timestamp value.

```
#include <lbmpdm.h>
```

### Data Fields

- int32\_t [tv\\_secs](#)
- int32\_t [tv\\_usecs](#)

#### 7.109.1 Detailed Description

The tv\_secs is the number of seconds since the epoch. The tv\_usecs is the additional microseconds since the epoch. and can range from -128 to 127.

#### 7.109.2 Field Documentation

##### 7.109.2.1 int32\_t lbmpdm\_timestamp\_t::tv\_secs

Seconds since epoch

##### 7.109.2.2 int32\_t lbmpdm\_timestamp\_t::tv\_usecs

Microseconds since last second

The documentation for this struct was generated from the following file:

- [lbmpdm.h](#)

## 7.110 lbmsdm\_decimal\_t\_stct Struct Reference

Structure to hold a scaled decimal number. A scaled decimal number consists of a mantissa  $m$  and an exponent  $exp$ . It represents the value  $m \cdot 10^{exp}$ .

```
#include <lbmsdm.h>
```

### Data Fields

- int64\_t `mant`
- int8\_t `exp`

#### 7.110.1 Detailed Description

The mantissa is represented as a 64-bit signed integer. The exponent is represented as an 8-bit signed integer, and can range from -128 to 127.

#### 7.110.2 Field Documentation

##### 7.110.2.1 int64\_t lbmsdm\_decimal\_t\_stct::mant

Mantissa.

##### 7.110.2.2 int8\_t lbmsdm\_decimal\_t\_stct::exp

Exponent.

The documentation for this struct was generated from the following file:

- [lbmsdm.h](#)

## 7.111 ume\_block\_src\_t\_stct Struct Reference

Structure used to designate an UME Block source.

```
#include <umeblocksrc.h>
```

### Data Fields

- `lbt_src_t * src`
- `lbt_src_cb_proc appproc`
- `ume_sem_t stablelock`
- `ume_block_bitmap_t * bitmap`
- `void * clientd`
- `int maxretentionsz`
- `int err`
- `unsigned int last`
- `unsigned int first`

The documentation for this struct was generated from the following file:

- [umeblocksrc.h](#)

## 7.112 ume\_liveness\_receiving\_context\_t\_stct Struct Reference

Structure that holds the information about a receiving context.

```
#include <lbm.h>
```

### Data Fields

- lbm\_uint64\_t **regid**
- lbm\_uint64\_t **session\_id**
- int **flag**

#### 7.112.1 Detailed Description

A structure used to hold a receiving context's user rcv regid and session id. Source contexts use this information to track receiver liveness.

The documentation for this struct was generated from the following file:

- [lbt.h](#)

# Chapter 8

## LBM API File Documentation

### 8.1 Ibm.h File Reference

Ultra Messaging (UM) API.

```
#include <inttypes.h>
```

Include dependency graph for Ibm.h:



### Data Structures

- struct [Ibm\\_iovec\\_t\\_stct](#)

*Structure, struct iovec compatible, that holds information about buffers used for vectored sends.*

- struct [Ibm\\_ipv4\\_address\\_mask\\_t\\_stct](#)

*Structure that holds an IPv4 address and a CIDR style netmask.*

- struct [Ibm\\_timeval\\_t\\_stct](#)

*Structure that holds seconds and microseconds since midnight, Jan 1, 1970 UTC.*

- struct [Ibm\\_src\\_event\\_wakeup\\_t\\_stct](#)

*Structure that holds source wakeup event data.*

- struct [Ibm\\_src\\_event\\_flight\\_size\\_notification\\_t\\_stct](#)

*Structure that holds flight size notification event data.*

- struct [lbt\\_src\\_event\\_ume\\_registration\\_t\\_stct](#)  
*Structure that holds store registration information for the UMP source.*
- struct [lbt\\_src\\_event\\_ume\\_registration\\_ex\\_t\\_stct](#)  
*Structure that holds store registration information for the UMP source in an extended form.*
- struct [lbt\\_src\\_event\\_ume\\_registration\\_complete\\_ex\\_t\\_stct](#)  
*Structure that holds information for sources after registration is complete to all involved stores.*
- struct [lbt\\_src\\_event\\_ume\\_deregistration\\_ex\\_t\\_stct](#)  
*Structure that holds store deregistration information for the UMP source in an extended form.*
- struct [lbt\\_msg\\_ume\\_registration\\_t\\_stct](#)  
*Structure that holds store registration information for the UMP receiver.*
- struct [lbt\\_msg\\_ume\\_registration\\_ex\\_t\\_stct](#)  
*Structure that holds store registration information for the UM receiver in an extended form.*
- struct [lbt\\_msg\\_ume\\_registration\\_complete\\_ex\\_t\\_stct](#)  
*Structure that holds information for receivers after registration is complete to all involved stores.*
- struct [lbt\\_msg\\_ume\\_deregistration\\_ex\\_t\\_stct](#)  
*Structure that holds store deregistration information for the UM receiver in an extended form.*
- struct [lbt\\_src\\_event\\_ume\\_ack\\_info\\_t\\_stct](#)  
*Structure that holds ACK information for a given message.*
- struct [lbt\\_src\\_event\\_ume\\_ack\\_ex\\_info\\_t\\_stct](#)  
*Structure that holds ACK information for a given message in an extended form.*
- struct [lbt\\_flight\\_size\\_inflight\\_t\\_stct](#)  
*Structure that holds information for source total inflight messages and bytes.*
- struct [lbt\\_umq\\_index\\_info\\_t\\_stct](#)  
*Structure that holds information used for sending and receiving messages with UMQ indices.*

- struct [lbt\\_msg\\_umq\\_index\\_assignment\\_eligibility\\_stop\\_complete\\_ex\\_t\\_stct](#)  
*Structure that holds index assignment information for receivers.*
- struct [lbt\\_msg\\_umq\\_index\\_assigned\\_ex\\_t\\_stct](#)  
*Structure that holds beginning-of-index information for receivers.*
- struct [lbt\\_msg\\_umq\\_index\\_released\\_ex\\_t\\_stct](#)  
*Structure that holds end-of-index information for receivers.*
- struct [lbt\\_msg\\_umq\\_index\\_assignment\\_eligibility\\_start\\_complete\\_ex\\_t\\_stct](#)  
*Structure that holds index assignment information for receivers.*
- struct [lbt\\_umq\\_msg\\_total\\_lifetime\\_info\\_t\\_stct](#)  
*Structure that holds UMQ message total lifetime information.*
- union [lbt\\_hf\\_sequence\\_number\\_t\\_stct](#)  
*Structure to hold a hot failover sequence number.*
- struct [lbt\\_umq\\_msgid\\_t\\_stct](#)  
*Structure that holds information for UMQ messages that allows the message to be identified uniquely.*
- struct [lbt\\_umq\\_queue\\_application\\_set\\_t\\_stct](#)
- struct [lbt\\_umq\\_queue\\_topic\\_t\\_stct](#)  
*Structure that holds queue topic information and can be used as a handle to a queue topic.*
- struct [lbt\\_umq\\_queue\\_topic\\_status\\_t](#)  
*Struct containing extended asynchronous operation status information about a single UMQ topic.*
- struct [lbt\\_ctx\\_umq\\_queue\\_topic\\_list\\_info\\_t](#)  
*Struct containing an array of queue topics retrieved via lbt\_umq\_queue\_topic\_list.*
- struct [lbt\\_umq\\_queue\\_msg\\_status\\_t](#)  
*Struct containing extended asynchronous operation status information about a single UMQ message.*
- struct [lbt\\_rcv\\_umq\\_queue\\_msg\\_list\\_info\\_t](#)  
*Struct containing an array of UMQ messages listed via lbt\_rcv\_umq\_queue\_msg\_list.*

- struct [lmb\\_rcv\\_umq\\_queue\\_msg\\_retrieve\\_info\\_t](#)  
*Struct containing an array of UMQ messages retrieved via lmb\_rcv\_umq\_queue\_msg\_retrieve.*
- struct [lmb\\_async\\_operation\\_info\\_t](#)  
*Results struct returned via the user-specified asynchronous operation callback from any asynchronous API.*
- struct [lmb\\_async\\_operation\\_func\\_t](#)  
*Structure that holds information for asynchronous operation callbacks.*
- struct [lmb\\_src\\_send\\_ex\\_info\\_t\\_stct](#)  
*Structure that holds information for the extended send calls A structure used with UM sources that utilize the extended send calls to pass options.*
- struct [lmb\\_ume\\_rcv\\_regid\\_ex\\_func\\_info\\_t\\_stct](#)  
*Structure that holds information for UMP receiver registration ID application callbacks.*
- struct [lmb\\_src\\_event\\_sequence\\_number\\_info\\_t\\_stct](#)  
*Structure that holds sequence number information for a message sent by a source.*
- struct [lmb\\_ume\\_rcv\\_recovery\\_info\\_ex\\_func\\_info\\_t\\_stct](#)  
*Structure that holds information for UMP receiver recovery sequence number info application callbacks.*
- struct [lmb\\_src\\_event\\_umq\\_message\\_id\\_info\\_t\\_stct](#)  
*Structure that holds Message ID information for a message sent by a sending UMQ application.*
- struct [lmb\\_context\\_event\\_umq\\_registration\\_ex\\_t\\_stct](#)  
*Structure that holds queue registration information for the UMQ context in an extended form.*
- struct [lmb\\_context\\_event\\_umq\\_registration\\_complete\\_ex\\_t\\_stct](#)  
*Structure that holds information for contexts after registration is complete to all involved queue instances.*
- struct [lmb\\_src\\_event\\_umq\\_registration\\_complete\\_ex\\_t\\_stct](#)  
*Structure that holds information for sources after registration is complete to all involved queue instances.*
- struct [lmb\\_msg\\_umq\\_registration\\_complete\\_ex\\_t\\_stct](#)

*Structure that holds information for receivers after registration is complete to all involved queue instances.*

- struct [lbt\\_src\\_event\\_umq\\_stability\\_ack\\_info\\_ex\\_t\\_stct](#)

*Structure that holds UMQ ACK information for a given message in an extended form.*

- struct [lbt\\_msg\\_umq\\_deregistration\\_complete\\_ex\\_t\\_stct](#)

*Structure that holds information for receivers after they de-register from a queue.*

- struct [lbt\\_src\\_event\\_umq\\_ulb\\_receiver\\_info\\_ex\\_t\\_stct](#)

*Structure that holds UMQ ULB receiver information in an extended form.*

- struct [lbt\\_src\\_event\\_umq\\_ulb\\_message\\_info\\_ex\\_t\\_stct](#)

*Structure that holds UMQ ULB message information in an extended form.*

- struct [lbt\\_str\\_hash\\_func\\_t\\_stct](#)

- struct [lbt\\_str\\_hash\\_func\\_ex\\_t\\_stct](#)

*Structure that holds the hash function callback information.*

- struct [lbt\\_src\\_notify\\_func\\_t\\_stct](#)

*Structure that holds the callback for source notifications.*

- struct [lbt\\_wildcard\\_rcv\\_compare\\_func\\_t\\_stct](#)

*Structure that holds the application callback pattern type information for wildcard receivers.*

- struct [lbt\\_ume\\_rcv\\_regid\\_func\\_t\\_stct](#)

*Structure that holds the application callback for registration ID setting.*

- struct [lbt\\_ume\\_rcv\\_regid\\_ex\\_func\\_t\\_stct](#)

*Structure that holds the application callback for registration ID setting, extended form.*

- struct [lbt\\_ume\\_src\\_force\\_reclaim\\_func\\_t\\_stct](#)

*Structure that holds the application callback for forced reclamation notifications.*

- struct [lbt\\_mim\\_unrecloss\\_func\\_t\\_stct](#)

*Structure that holds the application callback for multicast immediate message unrecoverable loss notification.*

- struct [lbt\\_ume\\_rcv\\_recovery\\_info\\_ex\\_func\\_t\\_stct](#)

*Structure that holds the application callback for recovery sequence number information, extended form.*

- struct [lbm\\_ume\\_store\\_entry\\_t\\_stct](#)  
*Structure that holds information for a UMP store for configuration purposes.*
- struct [lbm\\_uicast\\_resolver\\_entry\\_t\\_stct](#)  
*Structure that holds information for a unicast resolver daemon for configuration purposes.*
- struct [lbm\\_ume\\_store\\_name\\_entry\\_t\\_stct](#)  
*Structure that holds information for a UMP store by name for configuration purposes.*
- struct [lbm\\_ume\\_store\\_group\\_entry\\_t\\_stct](#)  
*Structure that holds information for a UMP store group for configuration purposes.*
- struct [lbm\\_rcv\\_src\\_notification\\_func\\_t\\_stct](#)  
*Structure that holds the application callback for source status notifications for receivers.*
- struct [ume\\_liveness\\_receiving\\_context\\_t\\_stct](#)  
*Structure that holds the information about a receiving context.*
- struct [lbm\\_ume\\_ctx\\_rcv\\_ctx\\_notification\\_func\\_t\\_stct](#)  
*Structure that holds the application callback for receiving context status notifications for source context.*
- struct [lbm\\_umq\\_queue\\_entry\\_t\\_stct](#)  
*Structure that holds information for a UMQ queue registration ID for configuration purposes.*
- struct [lbm\\_umq\\_ulb\\_receiver\\_type\\_entry\\_t\\_stct](#)  
*Structure that holds information for a UMQ ULB sources receiver type associations with application sets.*
- struct [lbm\\_umq\\_ulb\\_application\\_set\\_attr\\_t\\_stct](#)  
*Structure that holds information for a UMQ ULB sources application set attributes.*
- struct [lbm\\_umq\\_ulb\\_receiver\\_type\\_attr\\_t\\_stct](#)  
*Structure that holds information for a UMQ ULB sources receiver type attributes.*
- struct [lbm\\_context\\_src\\_event\\_func\\_t\\_stct](#)  
*Structure that holds the application callback for context-level source events.*
- struct [lbm\\_context\\_event\\_func\\_t\\_stct](#)

*Structure that holds the application callback for context-level events.*

- struct [lbt\\_serialized\\_response\\_t\\_stct](#)  
*Structure that holds a serialized UM response object.*
- struct [lbt\\_msg\\_fragment\\_info\\_t\\_stct](#)  
*Structure that holds fragment information for UM messages when appropriate.*
- struct [lbt\\_msg\\_gateway\\_info\\_t\\_stct](#)  
*Structure that holds originating information for UM messages which arrived via a gateway.*
- struct [lbt\\_msg\\_channel\\_info\\_t\\_stct](#)  
*Structure that represents UMS Spectrum channel information.*
- struct [lbt\\_msg\\_t\\_stct](#)  
*Structure that stores information about a received message.*
- struct [lbt\\_context\\_rcv\\_immediate\\_msgs\\_func\\_t\\_stct](#)  
*Structure that holds the application callback for receiving topic-less immediate mode messages.*
- struct [lbt\\_transport\\_source\\_info\\_t\\_stct](#)  
*Structure that holds formatted and parsed transport source strings.*
- struct [lbt\\_src\\_cost\\_func\\_t\\_stct](#)  
*Structure that holds the "source\_cost\_evaluation\_function" context attribute.*
- struct [lbt\\_config\\_option\\_stct\\_t](#)
- struct [lbt\\_wildcard\\_rcv\\_create\\_func\\_t\\_stct](#)  
*Structure that holds the receiver creation callback information for wildcard receivers.*
- struct [lbt\\_wildcard\\_rcv\\_delete\\_func\\_t\\_stct](#)  
*Structure that holds the receiver deletion callback information for wildcard receivers.*
- struct [lbt\\_src\\_transport\\_stats\\_tcp\\_t\\_stct](#)  
*Structure that holds datagram statistics for source TCP transports.*
- struct [lbt\\_src\\_transport\\_stats\\_lbtrm\\_t\\_stct](#)  
*Structure that holds datagram statistics for source LBT-RM transports.*
- struct [lbt\\_src\\_transport\\_stats\\_daemon\\_t\\_stct](#)  
*Structure that holds statistics for source daemon mode transport (deprecated).*

- struct [lbm\\_src\\_transport\\_stats\\_lbtru\\_t\\_stct](#)  
*Structure that holds datagram statistics for source LBT-RU transports.*
- struct [lbm\\_src\\_transport\\_stats\\_lbtipc\\_t\\_stct](#)  
*Structure that holds datagram statistics for source LBT-IPC transports.*
- struct [lbm\\_src\\_transport\\_stats\\_lbtrdma\\_t\\_stct](#)  
*Structure that holds datagram statistics for source LBT-RDMA transports.*
- struct [lbm\\_src\\_transport\\_stats\\_t\\_stct](#)  
*Structure that holds statistics for source transports.*
- struct [lbm\\_rcv\\_transport\\_stats\\_tcp\\_t\\_stct](#)  
*Structure that holds datagram statistics for receiver TCP transports.*
- struct [lbm\\_rcv\\_transport\\_stats\\_lbtrm\\_t\\_stct](#)  
*Structure that holds datagram statistics for receiver LBT-RM transports.*
- struct [lbm\\_rcv\\_transport\\_stats\\_daemon\\_t\\_stct](#)  
*Structure that holds statistics for receiver daemon mode transport (deprecated).*
- struct [lbm\\_rcv\\_transport\\_stats\\_lbtru\\_t\\_stct](#)  
*Structure that holds datagram statistics for receiver LBT-RU transports.*
- struct [lbm\\_rcv\\_transport\\_stats\\_lbtipc\\_t\\_stct](#)  
*Structure that holds datagram statistics for receiver LBT-IPC transports.*
- struct [lbm\\_rcv\\_transport\\_stats\\_lbtrdma\\_t\\_stct](#)  
*Structure that holds datagram statistics for receiver LBT-RDMA transports.*
- struct [lbm\\_rcv\\_transport\\_stats\\_t\\_stct](#)  
*Structure that holds statistics for receiver transports.*
- struct [lbm\\_event\\_queue\\_stats\\_t\\_stct](#)  
*Structure that holds statistics for an event queue.*
- struct [lbm\\_context\\_stats\\_t\\_stct](#)  
*Structure that holds statistics for a context.*
- struct [lbm\\_event\\_queue\\_cancel\\_cb\\_info\\_t\\_stct](#)  
*Structure passed to cancel/delete functions so that a cancel callback may be called.*

- struct [lbt\\_apphdr\\_chain\\_elem\\_t\\_stct](#)  
*Structure that represents an element in an app header chain.*
- struct [lbt\\_msg\\_properties\\_iter\\_t\\_stct](#)  
*A struct used for iterating over properties pointed to by an lbt\_msg\_properties\_t.*
- struct [lbt\\_umm\\_info\\_t\\_stct](#)  
*Structure for specifying UMM daemon connection options.*

## Defines

- #define **LBM\_VERS\_MAJOR** 5
- #define **LBM\_VERS\_MINOR** 3
- #define **LBM\_VERS\_MAINT** 6
- #define **LBM\_VERS\_SFX** 0
- #define **LBM\_VERS\_TAG** ""
- #define **LBM\_VERS** (LBM\_VERS\_MAJOR\*10000+LBM\_VERS\_-MINOR\*100+LBM\_VERS\_MAINT)
- #define **PRIuSZ** "zu"
- #define **PRIuSZcast(x)** (size\_t)(x)
- #define **SCNuSZ** "zu"
- #define **SCNuSZcast(x)** (size\_t \* )(x)
- #define **LBMEExpDLL**
- #define **LBM\_EINVAL** 1
- #define **LBM\_EWOULDBLOCK** 2
- #define **LBM\_ENOMEM** 3
- #define **LBM\_EOP** 4
- #define **LBM\_EOS** 5
- #define **LBM\_ETIMEDOUT** 6
- #define **LBM\_EDAEMONCONN** 7
- #define **LBM\_EUMENOREG** 8
- #define **LBM\_EOPNOTSUPP** 9
- #define **LBM\_EINPROGRESS** 10
- #define **LBM\_ENO\_QUEUE\_REG** 11
- #define **LBM\_ENO\_STORE\_REG** 12
- #define **LBM\_EMSG\_SELECTOR** 14
- #define **LBM\_MSG\_DATA** 0
- #define **LBM\_MSG\_EOS** 1
- #define **LBM\_MSG\_REQUEST** 2
- #define **LBM\_MSG\_RESPONSE** 3

- #define LBM\_MSG\_UNRECOVERABLE LOSS 4
- #define LBM\_MSG\_UNRECOVERABLE LOSS\_BURST 5
- #define LBM\_MSG\_NO\_SOURCE\_NOTIFICATION 6
- #define LBM\_MSG\_UME\_REGISTRATION\_ERROR 7
- #define LBM\_MSG\_UME\_REGISTRATION\_SUCCESS 8
- #define LBM\_MSG\_UME\_REGISTRATION\_CHANGE 9
- #define LBM\_MSG\_UME\_REGISTRATION\_SUCCESS\_EX 10
- #define LBM\_MSG\_UME\_REGISTRATION\_COMPLETE\_EX 11
- #define LBM\_MSG\_UME\_DEREGISTRATION\_SUCCESS\_EX 12
- #define LBM\_MSG\_UME\_DEREGISTRATION\_COMPLETE\_EX 13
- #define LBM\_MSG\_UMQ\_REGISTRATION\_ERROR 16
- #define LBM\_MSG\_UMQ\_REGISTRATION\_COMPLETE\_EX 18
- #define LBM\_MSG\_UMQ\_DEREGISTRATION\_COMPLETE\_EX 19
- #define LBM\_MSG\_BOS 20
- #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNMENT\_ELIGIBILITY\_ERROR 21
- #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNMENT\_ELIGIBILITY\_START\_COMPLETE\_EX 22
- #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNMENT\_ELIGIBILITY\_STOP\_COMPLETE\_EX 23
- #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNED\_EX 24
- #define LBM\_MSG\_UMQ\_INDEX\_RELEASED\_EX 25
- #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNMENT\_ERROR 26
- #define LBM\_MSG\_HF\_RESET 27
- #define LBM\_MSG\_START\_BATCH 0x1
- #define LBM\_MSG\_END\_BATCH 0x2
- #define LBM\_MSG\_COMPLETE\_BATCH 0x3
- #define LBM\_MSG\_FLUSH 0x4
- #define LBM\_SRC\_NONBLOCK 0x8
- #define LBM\_SRC\_BLOCK\_TEMP 0x10
- #define LBM\_SRC\_BLOCK 0x20
- #define LBM\_MSG iov\_GATHER 0x40
- #define LBM\_RCV\_NONBLOCK 0x8
- #define LBM\_RCV\_BLOCK\_TEMP 0x10
- #define LBM\_RCV\_BLOCK 0x20
- #define LBM\_MSG\_FLAG\_START\_BATCH 0x1
- #define LBM\_MSG\_FLAG\_END\_BATCH 0x2
- #define LBM\_MSG\_FLAG\_HF\_PASS\_THROUGH 0x4
- #define LBM\_MSG\_FLAG\_UME\_RETRANSMIT 0x8
- #define LBM\_MSG\_FLAG\_RETRANSMIT 0x8
- #define LBM\_MSG\_FLAG\_IMMEDIATE 0x10
- #define LBM\_MSG\_FLAG\_HF\_DUPLICATE 0x20
- #define LBM\_MSG\_FLAG\_UMQ\_REASSIGNED 0x40

- #define LBM\_MSG\_FLAG\_UMQ\_RESUBMITTED 0x80
- #define LBM\_MSG\_FLAG\_TOPICLESS 0x100
- #define LBM\_MSG\_FLAG\_DELIVERY\_LATENCY 0x200
- #define LBM\_MSG\_FLAG\_HF\_OPTIONAL 0x400
- #define LBM\_MSG\_FLAG\_HF\_32 0x800
- #define LBM\_MSG\_FLAG\_HF\_64 0x1000
- #define LBM\_MSG\_FLAG\_OTR 0x2000
- #define LBM\_MSG\_FLAG\_NUMBERED\_CHANNEL 0x1
- #define LBM\_TOPIC\_RES\_REQUEST\_RESERVED1 0x08
- #define LBM\_TOPIC\_RES\_REQUEST\_ADVERTISEMENT 0x04
- #define LBM\_TOPIC\_RES\_REQUEST\_QUERY 0x02
- #define LBM\_TOPIC\_RES\_REQUEST\_WILDCARD\_QUERY 0x01
- #define LBM\_SRC\_EVENT\_CONNECT 1
- #define LBM\_SRC\_EVENT\_DISCONNECT 2
- #define LBM\_SRC\_EVENT\_WAKEUP 3
- #define LBM\_SRC\_EVENT\_DAEMON\_CONFIRM 4
- #define LBM\_SRC\_EVENT\_UME\_REGISTRATION\_ERROR 5
- #define LBM\_SRC\_EVENT\_UME\_REGISTRATION\_SUCCESS 6
- #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE 7
- #define LBM\_SRC\_EVENT\_UME\_DELIVERY\_CONFIRMATION 8
- #define LBM\_SRC\_EVENT\_UME\_STORE\_UNRESPONSIVE 9
- #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_RECLAIMED 10
- #define LBM\_SRC\_EVENT\_UME\_REGISTRATION\_SUCCESS\_EX 11
- #define LBM\_SRC\_EVENT\_UME\_REGISTRATION\_COMPLETE\_EX 12
- #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE\_EX 13
- #define LBM\_SRC\_EVENT\_UME\_DELIVERY\_CONFIRMATION\_EX 14
- #define LBM\_SRC\_EVENT\_SEQUENCE\_NUMBER\_INFO 15
- #define LBM\_SRC\_EVENT\_UMQ\_REGISTRATION\_ERROR 16
- #define LBM\_SRC\_EVENT\_UMQ\_MESSAGE\_ID\_INFO 17
- #define LBM\_SRC\_EVENT\_UMQ\_REGISTRATION\_COMPLETE\_EX 18
- #define LBM\_SRC\_EVENT\_UMQ\_MESSAGE\_STABLE\_EX 19
- #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_ASSIGNED\_EX 20
- #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_REASSIGNED\_EX 21
- #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_TIMEOUT\_EX 22
- #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_COMPLETE\_EX 23
- #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_CONSUMED\_EX 24
- #define LBM\_SRC\_EVENT\_UMQ\_ULB\_RECEIVER\_REGISTRATION\_EX 25
- #define LBM\_SRC\_EVENT\_UMQ\_ULB\_RECEIVER\_DEREGISTRATION\_EX 26
- #define LBM\_SRC\_EVENT\_UMQ\_ULB\_RECEIVER\_READY\_EX 27
- #define LBM\_SRC\_EVENT\_UMQ\_ULB\_RECEIVER\_TIMEOUT\_EX 28
- #define LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION 29

- #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_RECLAIMED\_EX 30
- #define LBM\_SRC\_EVENT\_UME\_DEREGISTRATION\_SUCCESS\_EX 31
- #define LBM\_SRC\_EVENT\_UME\_DEREGISTRATION\_COMPLETE\_EX 32
- #define LBM\_CONTEXT\_EVENT\_UMQ\_REGISTRATION\_COMPLETE\_EX 1
- #define LBM\_CONTEXT\_EVENT\_UMQ\_REGISTRATION\_SUCCESS\_EX 2
- #define LBM\_CONTEXT\_EVENT\_UMQ\_REGISTRATION\_ERROR 3
- #define LBM\_CONTEXT\_EVENT\_UMQ\_INSTANCE\_LIST\_NOTIFICATION 4
- #define LBM\_TRANSPORT\_TYPE\_TCP 0x00
- #define LBM\_TRANSPORT\_TYPE\_LBTRU 0x01
- #define LBM\_TRANSPORT\_TYPE\_LBTRM 0x10
- #define LBM\_TRANSPORT\_TYPE\_LBTIPC 0x40
- #define LBM\_TRANSPORT\_TYPE\_LBTRDMA 0x20
- #define LBM\_CTX\_ATTR\_OP\_EMBEDDED 1
- #define LBM\_CTX\_ATTR\_OP\_DAEMON 2
- #define LBM\_CTX\_ATTR\_OP\_SEQUENTIAL 3
- #define LBM\_CTX\_ATTR\_FDTYPE\_POLL 1
- #define LBM\_CTX\_ATTR\_FDTYPE\_SELECT 2
- #define LBM\_CTX\_ATTR\_FDTYPE\_WSAEV 3
- #define LBM\_CTX\_ATTR\_FDTYPE\_WINCPOR 4
- #define LBM\_CTX\_ATTR\_FDTYPE\_WINCPOR\_OV 5
- #define LBM\_CTX\_ATTR\_FDTYPE\_EPOLL 6
- #define LBM\_CTX\_ATTR\_FDTYPE\_DEVPOLL 7
- #define LBM\_CTX\_ATTR\_FDTYPE\_KQUEUE 8
- #define LBM\_CTX\_ATTR\_FDTYPE\_WINRIOPOR 9
- #define LBM\_CTX\_ATTR\_MON\_TRANSPORT\_LBM 1
- #define LBM\_CTX\_ATTR\_MON\_TRANSPORT\_LBMSNMP 2
- #define LBM\_CTX\_ATTR\_IPC\_RCV\_THREAD\_PEND 1
- #define LBM\_CTX\_ATTR\_IPC\_RCV\_THREAD\_BUSY\_WAIT 2
- #define LBM\_CTX\_ATTR\_RDMA\_RCV\_THREAD\_PEND 1
- #define LBM\_CTX\_ATTR\_RDMA\_RCV\_THREAD\_BUSY\_WAIT 2
- #define LBM\_CTX\_ATTR\_RCV\_THREAD\_POOL\_CREATE 1
- #define LBM\_CTX\_ATTR\_RCV\_THREAD\_POOL\_DYNAMIC 2
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_DEFAULT 0x00000000
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_3\_6 0x00030600
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_3\_6\_1 0x00030601
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_3\_6\_2 0x00030602

- #define **5** 0x00030605 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_3\_6\_-
- #define **0** 0x00040000 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_-
- #define **1** 0x00040001 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_0\_-
- #define **1** 0x00040100 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_-
- #define **1** 0x00040101 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_1\_-
- #define **2** 0x00040102 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_1\_-
- #define **3** 0x00040103 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_1\_-
- #define **1** 0x00040201 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_2\_-
- #define **2** 0x00040202 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_2\_-
- #define **3** 0x00040203 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_2\_-
- #define **4** 0x00040204 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_2\_-
- #define **5** 0x00040205 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_2\_-
- #define **6** 0x00040206 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_2\_-
- #define **7** 0x00040207 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_2\_-
- #define **8** 0x00040208 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_2\_-
- #define **0** 0x01030000 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UME\_3\_-
- #define **1** 0x01030001 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UME\_3\_0\_-
- #define **2** 0x01030002 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UME\_3\_0\_-
- #define **1** 0x01030100 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UME\_3\_-
- #define **1** 0x01030101 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UME\_3\_1\_-
- #define **2** 0x01030102 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UME\_3\_1\_-
- #define **3** 0x01030103 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UME\_3\_1\_-

- #define **1** 0x01030201  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UME\_3\_2\_-**
- #define **2** 0x01030202  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UME\_3\_2\_-**
- #define **3** 0x01030203  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UME\_3\_2\_-**
- #define **4** 0x01030204  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UME\_3\_2\_-**
- #define **5** 0x01030205  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UME\_3\_2\_-**
- #define **6** 0x01030206  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UME\_3\_2\_-**
- #define **7** 0x01030207  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UME\_3\_2\_-**
- #define **8** 0x01030208  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UME\_3\_2\_-**
- #define **0** 0x02010000  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_1\_-**
- #define **1** 0x02010100  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_1\_-**
- #define **1** 0x02010101  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_1\_1\_-**
- #define **0** 0x02020000  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_-**
- #define **1** 0x02020001  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_0\_-**
- #define **1** 0x02020101  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_1\_-**
- #define **3** 0x02020103  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_1\_-**
- #define **4** 0x02020104  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_1\_-**
- #define **5** 0x02020105  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_1\_-**
- #define **6** 0x02020106  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_1\_-**
- #define **7** 0x02020107  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_1\_-**
- #define **8** 0x02020108  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_1\_-**
- #define **9** 0x02020109  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_1\_-**
- #define **10** 0x0202010a  
**LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_1\_-**

- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UM\_5\_0 0x03050000
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UM\_5\_0\_-  
1 0x03050001
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UM\_5\_1 0x03050100
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UM\_5\_1\_-  
1 0x03050101
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UM\_5\_1\_-  
2 0x03050102
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UM\_5\_2 0x03050200
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UM\_5\_2\_-  
1 0x03050201
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UM\_5\_2\_-  
2 0x03050202
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UM\_5\_3 0x03050300
- #define LBM\_SRC\_TOPIC\_ATTR\_TRANSPORT\_TCP LBM\_-  
TRANSPORT\_TYPE\_TCP
- #define LBM\_SRC\_TOPIC\_ATTR\_TRANSPORT\_LBTRM LBM\_-  
TRANSPORT\_TYPE\_LBTRM
- #define LBM\_SRC\_TOPIC\_ATTR\_TRANSPORT\_LBTRU LBM\_-  
TRANSPORT\_TYPE\_LBTRU
- #define LBM\_SRC\_TOPIC\_ATTR\_TRANSPORT\_LBTIPC LBM\_-  
TRANSPORT\_TYPE\_LBTIPC
- #define LBM\_SRC\_TOPIC\_ATTR\_TRANSPORT\_LBTRDMA LBM\_-  
TRANSPORT\_TYPE\_LBTRDMA
- #define LBM\_SRC\_TOPIC\_ATTR\_TCP\_MULTI\_RECV\_NORMAL 0
- #define LBM\_SRC\_TOPIC\_ATTR\_TCP\_MULTI\_RECV\_BOUNDED\_-  
LATENCY 1
- #define LBM\_SRC\_TOPIC\_ATTR\_TCP\_MULTI\_RECV\_SOURCE\_-  
PACED 2
- #define LBM\_SRC\_TOPIC\_ATTR\_TCP\_MULTI\_RECV\_SEND\_-  
ORDER\_SERIAL 0
- #define LBM\_SRC\_TOPIC\_ATTR\_TCP\_MULTI\_RECV\_SEND\_-  
ORDER\_RANDOM 1
- #define LBM\_SRC\_TOPIC\_ATTR\_SSF\_NONE 0
- #define LBM\_SRC\_TOPIC\_ATTR\_SSF\_INCLUSION 1
- #define LBM\_SRC\_TOPIC\_ATTR\_SSF\_EXCLUSION 2
- #define LBM\_SRC\_TOPIC\_ATTR\_IMPLICIT\_BATCH\_TYPE\_-  
DEFAULT 0
- #define LBM\_SRC\_TOPIC\_ATTR\_IMPLICIT\_BATCH\_TYPE\_-  
ADAPTIVE 1
- #define LBM\_SRC\_TOPIC\_ATTR\_UME\_STORE\_BEHAVIOR\_RR 0x0
- #define LBM\_SRC\_TOPIC\_ATTR\_UME\_STORE\_BEHAVIOR\_QC 0x1
- #define LBM\_SRC\_TOPIC\_ATTR\_UME\_QC\_SQN\_BEHAVIOR\_-  
LOWEST 0x0

- #define **LBM\_SRC\_TOPIC\_ATTR\_UME\_QC\_SQN\_BEHAVIOR\_MAJORITY** 0x1
- #define **LBM\_SRC\_TOPIC\_ATTR\_UME\_QC\_SQN\_BEHAVIOR\_HIGHEST** 0x2
- #define **LBM\_SRC\_TOPIC\_ATTR\_UME\_STABLE\_BEHAVIOR\_ANY** 0x0
- #define **LBM\_SRC\_TOPIC\_ATTR\_UME\_STABLE\_BEHAVIOR\_MAJORITY** 0x1
- #define **LBM\_SRC\_TOPIC\_ATTR\_UME\_STABLE\_BEHAVIOR\_QUORUM** 0x1
- #define **LBM\_SRC\_TOPIC\_ATTR\_UME\_STABLE\_BEHAVIOR\_ALL** 0x2
  
- #define **LBM\_SRC\_TOPIC\_ATTR\_UME\_STABLE\_BEHAVIOR\_ALL\_ACTIVE** 0x3
- #define **LBM\_SRC\_TOPIC\_ATTR\_UMQ\_STABLE\_BEHAVIOR\_ANY** 0x0
- #define **LBM\_SRC\_TOPIC\_ATTR\_UMQ\_STABLE\_BEHAVIOR\_MAJORITY** 0x1
- #define **LBM\_SRC\_TOPIC\_ATTR\_UMQ\_STABLE\_BEHAVIOR\_QUORUM** 0x1
- #define **LBM\_SRC\_TOPIC\_ATTR\_UMQ\_STABLE\_BEHAVIOR\_ALL** 0x2
- #define **LBM\_SRC\_TOPIC\_ATTR\_UMQ\_STABLE\_BEHAVIOR\_ALL\_ACTIVE** 0x3
- #define **LBM\_SRC\_TOPIC\_ATTR\_LBTIPC\_BEHAVIOR\_SOURCE\_PACED** LBTIPC\_BEHAVIOR\_SRC\_PACED
- #define **LBM\_SRC\_TOPIC\_ATTR\_LBTIPC\_BEHAVIOR\_RECEIVER\_PACED** LBTIPC\_BEHAVIOR\_RCVR\_PACED
- #define **LBM\_SRC\_TOPIC\_ATTR\_UMQ\_ULB\_EVENT\_MSG\_CONSUME** 0x1
- #define **LBM\_SRC\_TOPIC\_ATTR\_UMQ\_ULB\_EVENT\_MSG\_TIMEOUT** 0x2
- #define **LBM\_SRC\_TOPIC\_ATTR\_UMQ\_ULB\_EVENT\_MSG\_ASSIGNMENT** 0x4
- #define **LBM\_SRC\_TOPIC\_ATTR\_UMQ\_ULB\_EVENT\_MSG\_REASSIGNMENT** 0x8
- #define **LBM\_SRC\_TOPIC\_ATTR\_UMQ\_ULB\_EVENT\_MSG\_COMPLETE** 0x10
- #define **LBM\_SRC\_TOPIC\_ATTR\_UMQ\_ULB\_EVENT\_RCV\_TIMEOUT** 0x20
- #define **LBM\_SRC\_TOPIC\_ATTR\_UMQ\_ULB\_EVENT\_RCV\_REGISTRATION** 0x40
- #define **LBM\_SRC\_TOPIC\_ATTR\_UMQ\_ULB\_EVENT\_RCV\_DEREGISTRATION** 0x80

- #define **LBM\_SRC\_TOPIC\_ATTR\_UMQ\_ULB\_EVENT\_RCV\_READY** 0x100
- #define **LBM\_SRC\_TOPIC\_ATTR\_UMQ\_ULB\_EVENT\_ALL** 0x1FF
- #define **LBM\_SRC\_TOPIC\_ATTR\_UMQ\_ULB\_ASSIGNMENT\_DEFAULT** 0x0
- #define **LBM\_SRC\_TOPIC\_ATTR\_UMQ\_ULB\_ASSIGNMENT\_RANDOM** 0x1
- #define **LBM\_SRC\_TOPIC\_ATTR\_UMQ\_ULB\_LF\_BEHAVIOR\_IGNORED** 0x0
- #define **LBM\_SRC\_TOPIC\_ATTR\_UMQ\_ULB\_LF\_BEHAVIOR\_PROVISIONED** 0x1
- #define **LBM\_SRC\_TOPIC\_ATTR\_UMQ\_ULB\_LF\_BEHAVIOR\_DYNAMIC** 0x2
- #define **LBM\_SRC\_TOPIC\_ATTR\_UME\_STABLE\_EVENT\_NONE** 0x0
- #define **LBM\_SRC\_TOPIC\_ATTR\_UME\_STABLE\_EVENT\_PER\_FRAGMENT** 0x1
- #define **LBM\_SRC\_TOPIC\_ATTR\_UME\_STABLE\_EVENT\_PER\_MESSAGE** 0x2
- #define **LBM\_SRC\_TOPIC\_ATTR\_UME\_STABLE\_EVENT\_FRAG\_AND\_MSG** 0x3
- #define **LBM\_SRC\_TOPIC\_ATTR\_UME\_CDELV\_EVENT\_NONE** 0x0
- #define **LBM\_SRC\_TOPIC\_ATTR\_UME\_CDELV\_EVENT\_PER\_FRAGMENT** 0x1
- #define **LBM\_SRC\_TOPIC\_ATTR\_UME\_CDELV\_EVENT\_PER\_MESSAGE** 0x2
- #define **LBM\_SRC\_TOPIC\_ATTR\_UME\_CDELV\_EVENT\_FRAG\_AND\_MSG** 0x3
- #define **LBM\_RCV\_TOPIC\_ATTR\_UME\_QC\_SQN\_BEHAVIOR\_LOWEST** 0x0
- #define **LBM\_RCV\_TOPIC\_ATTR\_UME\_QC\_SQN\_BEHAVIOR\_MAJORITY** 0x1
- #define **LBM\_RCV\_TOPIC\_ATTR\_UME\_QC\_SQN\_BEHAVIOR\_HIGHEST** 0x2
- #define **LBM\_RCV\_TOPIC\_ATTR\_TCP\_ACTIVITY\_TIMEOUT\_SO\_KEEPALIVE** 0x1
- #define **LBM\_RCV\_TOPIC\_ATTR\_TCP\_ACTIVITY\_TIMEOUT\_TIMER** 0x2
- #define **LBM\_RCV\_TOPIC\_ATTR\_CHANNEL\_BEHAVIOR\_DELIVER\_MSGS** 0x1
- #define **LBM\_RCV\_TOPIC\_ATTR\_CHANNEL\_BEHAVIOR\_DISCARD\_MSGS** 0x2
- #define **LBM\_RCV\_TOPIC\_ATTR\_UMQ\_INDEX\_ASSIGN\_ELIGIBILITY\_INELIGIBLE** 0x0

- #define **LBM\_RCV\_TOPIC\_ATTR\_UMQ\_INDEX\_ASSIGN\_ELIGIBILITY\_ELIGIBLE** 0x1
- #define **LBM\_RCV\_TOPIC\_ATTR\_UMQ\_QUEUE\_PARTICIPATION\_NONE** 0
- #define **LBM\_RCV\_TOPIC\_ATTR\_UMQ\_QUEUE\_PARTICIPATION\_NORMAL** 1
- #define **LBM\_RCV\_TOPIC\_ATTR\_UMQ\_QUEUE\_PARTICIPATION\_OBSERVER** 2
- #define **LBM\_RCV\_TOPIC\_ATTR\_UMQ\_HOLD\_INTERVAL\_FOREVER** 0xFFFFFFFF
- #define **LBM\_MSG\_MAX\_SOURCE\_LEN** 128
- #define **LBM\_MSG\_MAX\_TOPIC\_LEN** 256
- #define **LBM\_MSG\_MAX\_STATE\_LEN** 32
- #define **LBM\_UME\_MAX\_STORE\_STRLEN** 24
- #define **LBM\_UMQ\_MAX\_QUEUE\_STRLEN** 256
- #define **LBM\_UMQ\_MAX\_TOPIC\_STRLEN** 256
- #define **LBM\_UMQ\_MAX\_APPSET\_STRLEN** 256
- #define **LBM\_MAX\_CONTEXT\_NAME\_LEN** 128
- #define **LBM\_MAX\_EVENT\_QUEUE\_NAME\_LEN** 128
- #define **LBM\_UMQ\_ULB\_MAX\_RECEIVER\_STRLEN** 32
- #define **LBM\_UMQ\_MAX\_INDEX\_LEN** 216
- #define **LBM\_UMQ\_USER\_NAME\_LENGTH\_MAX** 127
- #define **LBM\_UMQ\_PASSWORD\_LENGTH\_MAX** 15
- #define **LBM\_UMM\_NUM\_SERVERS\_MAX** 16
- #define **LBM\_UMM\_USER\_NAME\_LENGTH\_MAX** 100
- #define **LBM\_UMM\_APP\_NAME\_LENGTH\_MAX** 100
- #define **LBM\_UMM\_PASSWORD\_LENGTH\_MAX** 100
- #define **LBM\_UMM\_SERVER\_LENGTH\_MAX** 32
- #define **LBM\_HMAC\_BLOCK\_SZ** 20
- #define **LBM\_FLIGHT\_SIZE\_BEHAVIOR\_NOTIFY** 0x0
- #define **LBM\_FLIGHT\_SIZE\_BEHAVIOR\_BLOCK** 0x1
- #define **LBM\_FD\_EVENT\_READ** 0x1
- #define **LBM\_FD\_EVENT\_WRITE** 0x2
- #define **LBM\_FD\_EVENT\_EXCEPT** 0x4
- #define **LBM\_FD\_EVENT\_ACCEPT** 0x8
- #define **LBM\_FD\_EVENT\_CLOSE** 0x10
- #define **LBM\_FD\_EVENT\_CONNECT** 0x20
- #define **LBM\_FD\_EVENT\_ALL** 0x3f
- #define **LBM\_EVENT\_QUEUE\_BLOCK** 0xFFFFFFFF
- #define **LBM\_EVENT\_QUEUE\_POLL** 0x0
- #define **LBM\_EVENT\_QUEUE\_SIZE\_WARNING** 0x1
- #define **LBM\_EVENT\_QUEUE\_DELAY\_WARNING** 0x2
- #define **LBM\_EVENT\_QUEUE\_ENQUEUE\_NOTIFICATION** 0x3

- #define LBM\_LOG\_EMERG 1
- #define LBM\_LOG\_ALERT 2
- #define LBM\_LOG\_CRIT 3
- #define LBM\_LOG\_ERR 4
- #define LBM\_LOG\_WARNING 5
- #define LBM\_LOG\_NOTICE 6
- #define LBM\_LOG\_INFO 7
- #define LBM\_LOG\_DEBUG 8
- #define LBM\_DAEMON\_EVENT\_CONNECTED 1
- #define LBM\_DAEMON\_EVENT\_CONNECT\_ERROR 2
- #define LBM\_DAEMON\_EVENT\_DISCONNECTED 3
- #define LBM\_DAEMON\_EVENT\_CONNECT\_TIMEOUT 4
- #define LBM\_TRANSPORT\_STAT\_TCP LBM\_TRANSPORT\_TYPE\_TCP
- #define LBM\_TRANSPORT\_STAT\_LBTRM LBM\_TRANSPORT\_TYPE\_LBTRM
- #define LBM\_TRANSPORT\_STAT\_DAEMON 0xFF
- #define LBM\_TRANSPORT\_STAT\_LBTRU LBM\_TRANSPORT\_TYPE\_LBTRU
- #define LBM\_TRANSPORT\_STAT\_LBTIPC LBM\_TRANSPORT\_TYPE\_LBTIPC
- #define LBM\_TRANSPORT\_STAT\_LBTRDMA LBM\_TRANSPORT\_TYPE\_LBTRDMA
- #define LBM\_WILDCARD\_RCV\_PATTERN\_TYPE\_PCRE 1
- #define LBM\_WILDCARD\_RCV\_PATTERN\_TYPE\_REGEX 2
- #define LBM\_WILDCARD\_RCV\_PATTERN\_TYPE\_APP\_CB 3
- #define LBM\_SRC\_EVENT\_WAKEUP\_FLAG\_NORMAL 0x1
- #define LBM\_SRC\_EVENT\_WAKEUP\_FLAG\_MIM 0x2
- #define LBM\_SRC\_EVENT\_WAKEUP\_FLAG\_UIM 0x4
- #define LBM\_SRC\_EVENT\_WAKEUP\_FLAG\_REQUEST 0x8
- #define LBM\_SRC\_EVENT\_WAKEUP\_FLAG\_RESPONSE 0x10
- #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_TIMEOUT\_EX\_FLAG\_TOTAL\_LIFETIME\_EXPIRED 0x1
- #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_TIMEOUT\_EX\_FLAG\_EXPLICIT 0x2
- #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_TIMEOUT\_EX\_FLAG\_DISCARD 0x4
- #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_TIMEOUT\_EX\_FLAG\_MAX\_REASSIGNS 0x8
- #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_REASSIGNED\_EX\_FLAG\_EXPLICIT 0x1
- #define LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION\_TYPE\_UME 0x1
- #define LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION\_TYPE\_ULB 0x2

- #define LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION\_TYPE\_UMQ 0x3
- #define LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION\_STATE\_OVER 0x1
- #define LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION\_STATE\_UNDER 0x2
- #define LBM\_FLIGHT\_SIZE\_TYPE\_UME 0x1
- #define LBM\_FLIGHT\_SIZE\_TYPE\_ULB 0x2
- #define LBM\_FLIGHT\_SIZE\_TYPE\_UMQ 0x3
- #define LBM\_SRC\_SEND\_EX\_FLAG\_UME\_CLIENTD 0x1
- #define LBM\_SRC\_SEND\_EX\_FLAG\_UMQ\_CLIENTD 0x1
- #define LBM\_SRC\_SEND\_EX\_FLAG\_SEQUENCE\_NUMBER\_INFO 0x2
- #define LBM\_SRC\_SEND\_EX\_FLAG\_SEQUENCE\_NUMBER\_INFO\_FRAGONLY 0x4
- #define LBM\_SRC\_SEND\_EX\_FLAG\_APPHDR\_CHAIN 0x8
- #define LBM\_SRC\_SEND\_EX\_FLAG\_UMQ\_MESSAGE\_ID\_INFO 0x10
- #define LBM\_SRC\_SEND\_EX\_FLAG\_CHANNEL 0x20
- #define LBM\_SRC\_SEND\_EX\_FLAG\_UMQ\_INDEX 0x40
- #define LBM\_SRC\_SEND\_EX\_FLAG\_UMQ\_TOTAL\_LIFETIME 0x80
- #define LBM\_SRC\_SEND\_EX\_FLAG\_HF\_OPTIONAL 0x100
- #define LBM\_SRC\_SEND\_EX\_FLAG\_PROPERTIES 0x200
- #define LBM\_SRC\_SEND\_EX\_FLAG\_HF\_32 0x400
- #define LBM\_SRC\_SEND\_EX\_FLAG\_HF\_64 0x800
- #define LBM\_MSG\_PROPERTY\_NONE 0x0
- #define LBM\_MSG\_PROPERTY\_BOOLEAN 0x1
- #define LBM\_MSG\_PROPERTY\_BYTE 0x2
- #define LBM\_MSG\_PROPERTY\_SHORT 0x3
- #define LBM\_MSG\_PROPERTY\_INT 0x4
- #define LBM\_MSG\_PROPERTY\_LONG 0x5
- #define LBM\_MSG\_PROPERTY\_FLOAT 0x6
- #define LBM\_MSG\_PROPERTY\_DOUBLE 0x7
- #define LBM\_MSG\_PROPERTY\_STRING 0x8
- #define LBM\_MSG\_PROPERTIES\_MAX\_NAMELEN 250
- #define LBM\_CHAIN\_ELEM\_CHANNEL\_NUMBER 0x1
- #define LBM\_CHAIN\_ELEM\_HF\_SQN 0x2
- #define LBM\_CHAIN\_ELEM\_GW\_INFO 0x3
- #define LBM\_CHAIN\_ELEM\_APPHDR 0x4
- #define LBM\_CHAIN\_ELEM\_USER\_DATA 0x5
- #define LBM\_CHAIN\_ELEM\_PROPERTIES\_LENGTH 0x6
- #define LBM\_SRC\_EVENT\_UME\_REGISTRATION\_SUCCESS\_EX\_FLAG\_OLD 0x1
- #define LBM\_SRC\_EVENT\_UME\_REGISTRATION\_SUCCESS\_EX\_FLAG\_NOACKS 0x2

- #define LBM\_SRC\_EVENT\_UME\_REGISTRATION\_SUCCESS\_EX\_FLAG\_RPP 0x4
- #define LBM\_SRC\_EVENT\_UME\_REGISTRATION\_COMPLETE\_EX\_FLAG\_QUORUM 0x1
- #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE\_EX\_FLAG\_INTRAGROUP\_STABLE 0x1
- #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE\_EX\_FLAG\_INTEGRGROU\_STABLE 0x2
- #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE\_EX\_FLAG\_STABLE 0x4
- #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE\_EX\_FLAG\_STORE 0x8
- #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE\_EX\_FLAG\_WHOLE\_MESSAGE\_STABLE 0x10
- #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE\_EX\_FLAG\_USER 0x20
- #define LBM\_SRC\_EVENT\_UME\_DELIVERY\_CONFIRMATION\_EX\_FLAG\_UNIQUEACKS 0x1
- #define LBM\_SRC\_EVENT\_UME\_DELIVERY\_CONFIRMATION\_EX\_FLAG\_UREGID 0x2
- #define LBM\_SRC\_EVENT\_UME\_DELIVERY\_CONFIRMATION\_EX\_FLAG\_OOD 0x4
- #define LBM\_SRC\_EVENT\_UME\_DELIVERY\_CONFIRMATION\_EX\_FLAG\_EXACK 0x8
- #define LBM\_SRC\_EVENT\_UME\_DELIVERY\_CONFIRMATION\_EX\_FLAG\_WHOLE\_MESSAGE\_CONFIRMED 0x10
- #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_RECLAIMED\_EX\_FLAG\_FORCED 0x1
- #define LBM\_MSG\_UME\_REGISTRATION\_SUCCESS\_EX\_FLAG\_OLD 0x1
- #define LBM\_MSG\_UME\_REGISTRATION\_SUCCESS\_EX\_FLAG\_NOCACHE 0x2
- #define LBM\_MSG\_UME\_REGISTRATION\_SUCCESS\_EX\_FLAG\_RPP 0x4
- #define LBM\_MSG\_UME\_REGISTRATION\_COMPLETE\_EX\_FLAG\_QUORUM 0x1
- #define LBM\_MSG\_UME\_REGISTRATION\_COMPLETE\_EX\_FLAG\_RXREQMAX 0x2
- #define LBM\_MSG\_UME\_DEREGISTRATION\_SUCCESS\_EX\_FLAG\_RPP 0x1
- #define LBM\_CONTEXT\_EVENT\_UMQ\_REGISTRATION\_COMPLETE\_EX\_FLAG\_QUORUM 0x1
- #define LBM\_SRC\_EVENT\_UMQ\_REGISTRATION\_COMPLETE\_EX\_FLAG\_QUORUM 0x1

- #define LBM\_SRC\_EVENT\_UML\_MESSAGE\_STABLE\_EX\_FLAG\_INTRAGROUP\_STABLE 0x1
- #define LBM\_SRC\_EVENT\_UML\_MESSAGE\_STABLE\_EX\_FLAG\_INTEGRGUP\_STABLE 0x2
- #define LBM\_SRC\_EVENT\_UML\_MESSAGE\_STABLE\_EX\_FLAG\_STABLE 0x4
- #define LBM\_SRC\_EVENT\_UML\_MESSAGE\_STABLE\_EX\_FLAG\_USER 0x8
- #define LBM\_MSG\_UML\_REGISTRATION\_COMPLETE\_EX\_FLAG\_QUORUM 0x1
- #define LBM\_MSG\_UML\_REGISTRATION\_COMPLETE\_EX\_FLAG\_ULB 0x2
- #define LBM\_MSG\_UML\_DEREGISTRATION\_COMPLETE\_EX\_FLAG\_ULB 0x1
- #define LBM\_MSG\_UML\_INDEX\_ASSIGNED\_EX\_FLAG\_ULB 0x1
- #define LBM\_MSG\_UML\_INDEX\_ASSIGNED\_EX\_FLAG\_REQUESTED 0x2
- #define LBM\_MSG\_UML\_INDEX\_RELEASED\_EX\_FLAG\_ULB 0x1
- #define LBM\_MSG\_UML\_INDEX\_ASSIGNMENT\_ELIGIBILITY\_STOP\_COMPLETE\_EX\_FLAG\_ULB 0x1
- #define LBM\_MSG\_UML\_INDEX\_ASSIGNMENT\_ELIGIBILITY\_START\_COMPLETE\_EX\_FLAG\_ULB 0x1
- #define LBM\_MSG\_UML\_REASSIGN\_FLAG\_DISCARD 0x1
- #define LBM\_UME\_LIVENESS\_RECEIVER\_UNRESPONSIVE\_FLAG\_TMO 0x1
- #define LBM\_UME\_LIVENESS\_RECEIVER\_UNRESPONSIVE\_FLAG\_EOF 0x2
- #define LBM\_UMM\_INFO\_FLAGS\_USE\_SSL 0x1
- #define LBM\_TEXTMESSAGE 0;
- #define LBM\_BYTEMESSAGE 1;
- #define LBM\_MAPMESSAGE 2;
- #define LBM\_MESSAGE 3;
- #define LBM\_OBJECTMESSAGE 4;
- #define LBM\_STREAMMESSAGE 5;
- #define LBM\_JMSDeliveryMode "JMSDeliveryMode"
- #define LBM\_JMSExpiration "JMSExpiration"
- #define LBM\_JMSPriority "JMSPriority"
- #define LBM\_JMSMessageID "JMSMessageID"
- #define LBM\_JMSTimestamp "JMSTimestamp"
- #define LBM\_JMSCorrelationID "JMSCorrelationID"
- #define LBM\_JMSType "JMSType"
- #define LBM\_JMSRedelivered "JMSRedelivered"
- #define LBM\_LBMMMessageType "LBMMMessageType"

- #define **LBM\_JMSTopicType** "JMSTopicType"
- #define **LBM\_JMSReplyToName** "JMSReplyToName"
- #define **LBM\_JMSReplyToWildcard** "JMSReplyToWildcard"
- #define **LBM\_JMSReplyToType** "JMSReplyToType"
- #define **LBM\_EXTERNAL\_STRUCT\_FILL\_SIZE** (512)
- #define **LBM\_UMQ\_INDEX\_FLAG\_NUMERIC** 0x1
- #define **LBM\_UMQ\_QUEUE\_MSG\_STATUS\_UNKNOWN** 0
  - Queue message status; queue has no knowledge of the message.*
- #define **LBM\_UMQ\_QUEUE\_MSG\_STATUS\_UNASSIGNED** 1
  - Queue message status; message is currently enqueued but not yet assigned.*
- #define **LBM\_UMQ\_QUEUE\_MSG\_STATUS\_ASSIGNED** 2
  - Queue message status; message is currently assigned to a receiver but not yet consumed.*
- #define **LBM\_UMQ\_QUEUE\_MSG\_STATUS\_REASSIGNING** 3
  - Queue message status; message is waiting to be re-assigned to a different receiver.*
- #define **LBM\_UMQ\_QUEUE\_MSG\_STATUS\_CONSUMED** 4
  - Queue message status; message has been fully consumed and is no longer present in the queue.*
- #define **LBM\_ASYNC\_OP\_TYPE\_CTX\_UMQ\_QUEUE\_TOPIC\_LIST** 1
- #define **LBM\_ASYNC\_OP\_TYPE\_RCV\_UMQ\_QUEUE\_MSG\_LIST** 2
- #define **LBM\_ASYNC\_OP\_TYPE\_RCV\_UMQ\_QUEUE\_MSG\_RETRIEVE** 3
- #define **LBM\_ASYNC\_OP\_STATUS\_IN\_PROGRESS** 1
- #define **LBM\_ASYNC\_OP\_STATUS\_COMPLETE** 128
- #define **LBM\_ASYNC\_OP\_STATUS\_ERROR** 129
- #define **LBM\_ASYNC\_OP\_STATUS\_CANCELED** 130
- #define **LBM\_ASYNC\_OP\_INVALID\_HANDLE** 0
- #define **LBM\_ASYNC\_OPERATION\_CANCEL\_FLAG\_NONBLOCK** 0x1
- #define **LBM\_ASYNC\_OPERATION\_STATUS\_FLAG\_NONBLOCK** 0x1
- #define **LBM\_ASYNC\_OP\_INFO\_FLAG\_INLINE** 0x1
- #define **LBM\_ASYNC\_OP\_INFO\_FLAG\_FIRST** 0x2
- #define **LBM\_ASYNC\_OP\_INFO\_FLAG\_LAST** 0x4
- #define **LBM\_ASYNC\_OP\_INFO\_FLAG\_ONLY** (LBM\_ASYNC\_OP\_INFO\_FLAG\_FIRST | LBM\_ASYNC\_OP\_INFO\_FLAG\_LAST)
- #define **LBM\_SRC\_COST\_FUNCTION\_REJECT** 0xffffffff
- #define **LBM\_CONFIG\_OPTIONS\_STR\_LEN** 128

*Config option structure holding the option name and value via string types.*

- #define **lmb\_rcv\_retrieve\_all\_transport\_stats**(r, n, s) lmb\_rcv\_retrieve\_all\_transport\_stats\_ex(r,n,sizeof(**lmb\_rcv\_transport\_stats\_t**),s)
- #define **lmb\_context\_retrieve\_rcv\_transport\_stats**(c, n, s) lmb\_context\_retrieve\_rcv\_transport\_stats\_ex(c,n,sizeof(**lmb\_rcv\_transport\_stats\_t**),s)
- #define **lmb\_context\_retrieve\_src\_transport\_stats**(c, n, s) lmb\_context\_retrieve\_src\_transport\_stats\_ex(c,n,sizeof(**lmb\_src\_transport\_stats\_t**),s)

## TypeDefs

- typedef unsigned int **lmb\_uint\_t**
- typedef unsigned long int **lmb\_ulong\_t**
- typedef unsigned short int **lmb\_ushort\_t**
- typedef unsigned char **lmb\_uchar\_t**
- typedef uint8\_t **lmb\_uint8\_t**
- typedef uint16\_t **lmb\_uint16\_t**
- typedef uint32\_t **lmb\_uint32\_t**
- typedef uint64\_t **lmb\_uint64\_t**
- typedef int64\_t **lmb\_int64\_t**
- typedef int **lmb\_handle\_t**
- typedef **lmb\_iovec\_t\_stct lmb\_iovec\_t**  
*Structure, struct iovec compatible, that holds information about buffers used for vectored sends.*
- typedef **lmb\_ipv4\_address\_mask\_t\_stct lmb\_ipv4\_address\_mask\_t**  
*Structure that holds an IPv4 address and a CIDR style netmask.*
- typedef **lmb\_timeval\_t\_stct lmb\_timeval\_t**  
*Structure that holds seconds and microseconds since midnight, Jan 1, 1970 UTC.*
- typedef **lmb\_src\_event\_wakeup\_t\_stct lmb\_src\_event\_wakeup\_t**  
*Structure that holds source wakeup event data.*
- typedef **lmb\_src\_event\_flight\_size\_notification\_t\_stct lmb\_src\_event\_flight\_size\_notification\_t**  
*Structure that holds flight size notification event data.*
- typedef **lmb\_src\_event\_ume\_registration\_t\_stct lmb\_src\_event\_ume\_registration\_t**  
*Structure that holds store registration information for the UMP source.*
- typedef **lmb\_src\_event\_ume\_registration\_ex\_t\_stct lmb\_src\_event\_ume\_registration\_ex\_t**

*Structure that holds store registration information for the UMP source in an extended form.*

- **typedef lbt\_src\_event\_ume\_registration\_complete\_ex\_t\_stct lbt\_src\_event\_ume\_registration\_complete\_ex\_t**

*Structure that holds information for sources after registration is complete to all involved stores.*

- **typedef lbt\_src\_event\_ume\_deregistration\_ex\_t\_stct lbt\_src\_event\_ume\_deregistration\_ex\_t**

*Structure that holds store deregistration information for the UMP source in an extended form.*

- **typedef lbt\_msg\_ume\_registration\_t\_stct lbt\_msg\_ume\_registration\_t**

*Structure that holds store registration information for the UMP receiver.*

- **typedef lbt\_msg\_ume\_registration\_ex\_t\_stct lbt\_msg\_ume\_registration\_ex\_t**

*Structure that holds store registration information for the UM receiver in an extended form.*

- **typedef lbt\_msg\_ume\_registration\_complete\_ex\_t\_stct lbt\_msg\_ume\_registration\_complete\_ex\_t**

*Structure that holds information for receivers after registration is complete to all involved stores.*

- **typedef lbt\_msg\_ume\_deregistration\_ex\_t\_stct lbt\_msg\_ume\_deregistration\_ex\_t**

*Structure that holds store deregistration information for the UM receiver in an extended form.*

- **typedef lbt\_src\_event\_ume\_ack\_info\_t\_stct lbt\_src\_event\_ume\_ack\_info\_t**

*Structure that holds ACK information for a given message.*

- **typedef lbt\_src\_event\_ume\_ack\_ex\_info\_t\_stct lbt\_src\_event\_ume\_ack\_ex\_info\_t**

*Structure that holds ACK information for a given message in an extended form.*

- **typedef lbt\_flight\_size\_inflight\_t\_stct lbt\_flight\_size\_inflight\_t**

*Structure that holds information for source total inflight messages and bytes.*

- **typedef lbt\_src\_channel\_info\_t\_stct lbt\_src\_channel\_info\_t**

- **typedef lbt\_umq\_index\_info\_t\_stct lbt\_umq\_index\_info\_t**

*Structure that holds information used for sending and receiving messages with UMQ indices.*

- **typedef lbm\_msg\_umq\_index\_assignment\_eligibility\_stop\_complete\_ex\_t\_stct lbm\_msg\_umq\_index\_assignment\_eligibility\_stop\_complete\_ex\_t**

*Structure that holds index assignment information for receivers.*

- **typedef lbm\_msg\_umq\_index\_assigned\_ex\_t\_stct lbm\_msg\_umq\_index\_assigned\_ex\_t**

*Structure that holds beginning-of-index information for receivers.*

- **typedef lbm\_msg\_umq\_index\_released\_ex\_t\_stct lbm\_msg\_umq\_index\_released\_ex\_t**

*Structure that holds end-of-index information for receivers.*

- **typedef lbm\_msg\_umq\_index\_assignment\_eligibility\_start\_complete\_ex\_t\_stct lbm\_msg\_umq\_index\_assignment\_eligibility\_start\_complete\_ex\_t**

*Structure that holds index assignment information for receivers.*

- **typedef lbm\_apphdr\_chain\_t\_stct lbm\_apphdr\_chain\_t**

- **typedef lbm\_umq\_msg\_total\_lifetime\_info\_t\_stct lbm\_umq\_msg\_total\_lifetime\_info\_t**

*Structure that holds UMQ message total lifetime information.*

- **typedef lbm\_hf\_sequence\_number\_t\_stct lbm\_hf\_sequence\_number\_t**

*Structure to hold a hot failover sequence number.*

- **typedef lbm\_uint64\_t lbm\_umq\_regid\_t**

- **typedef lbm\_umq\_mgid\_t\_stct lbm\_umq\_mgid\_t**

*Structure that holds information for UMQ messages that allows the message to be identified uniquely.*

- **typedef lbm\_umq\_queue\_application\_set\_t\_stct lbm\_umq\_queue\_application\_set\_t**

- **typedef lbm\_umq\_queue\_topic\_t\_stct lbm\_umq\_queue\_topic\_t**

*Structure that holds queue topic information and can be used as a handle to a queue topic.*

- **typedef lbm\_msg\_t\_stct lbm\_msg\_t**

- **typedef lbm\_event\_queue\_t\_stct lbm\_event\_queue\_t**

- **typedef lbm\_uint64\_t lbm\_async\_operation\_handle\_t**

*Opaque handle to an asynchronous operation.*

- **typedef int(\*) lbt\_async\_operation\_function\_cb (lbt\_async\_operation\_info\_t \*opinfo, void \*clientd)**  
*User-supplied application callback for asynchronous operation status and completion.*
- **typedef lbt\_msg\_properties\_t\_stct lbt\_msg\_properties\_t**
- **typedef lbt\_src\_send\_ex\_info\_t\_stct lbt\_src\_send\_ex\_info\_t**  
*Structure that holds information for the extended send calls A structure used with UMP sources that utilize the extended send calls to pass options.*
- **typedef lbt\_ume\_rcv\_regid\_ex\_func\_info\_t\_stct lbt\_ume\_rcv\_regid\_ex\_func\_info\_t**  
*Structure that holds information for UMP receiver registration ID application callbacks.*
- **typedef lbt\_src\_event\_sequence\_number\_info\_t\_stct lbt\_src\_event\_sequence\_number\_info\_t**  
*Structure that holds sequence number information for a message sent by a source.*
- **typedef lbt\_ume\_rcv\_recovery\_info\_ex\_func\_info\_t\_stct lbt\_ume\_rcv\_recovery\_info\_ex\_func\_info\_t**  
*Structure that holds information for UMP receiver recovery sequence number info application callbacks.*
- **typedef lbt\_src\_event\_umq\_message\_id\_info\_t\_stct lbt\_src\_event\_umq\_message\_id\_info\_t**  
*Structure that holds Message ID information for a message sent by a sending UMQ application.*
- **typedef lbt\_context\_event\_umq\_registration\_ex\_t\_stct lbt\_context\_event\_umq\_registration\_ex\_t**  
*Structure that holds queue registration information for the UMQ context in an extended form.*
- **typedef lbt\_context\_event\_umq\_registration\_complete\_ex\_t\_stct lbt\_context\_event\_umq\_registration\_complete\_ex\_t**  
*Structure that holds information for contexts after registration is complete to all involved queue instances.*
- **typedef lbt\_src\_event\_umq\_registration\_complete\_ex\_t\_stct lbt\_src\_event\_umq\_registration\_complete\_ex\_t**  
*Structure that holds information for sources after registration is complete to all involved queue instances.*

- **typedef lbm\_msg\_umq\_registration\_complete\_ex\_t\_stct lbm\_msg\_umq\_registration\_complete\_ex\_t**  
*Structure that holds information for receivers after registration is complete to all involved queue instances.*
- **typedef lbm\_src\_event\_umq\_stability\_ack\_info\_ex\_t\_stct lbm\_src\_event\_umq\_stability\_ack\_info\_ex\_t**  
*Structure that holds UMQ ACK information for a given message in an extended form.*
- **typedef lbm\_msg\_umq\_deregistration\_complete\_ex\_t\_stct lbm\_msg\_umq\_deregistration\_complete\_ex\_t**  
*Structure that holds information for receivers after they de-register from a queue.*
- **typedef lbm\_src\_event\_umq\_ulb\_receiver\_info\_ex\_t\_stct lbm\_src\_event\_umq\_ulb\_receiver\_info\_ex\_t**  
*Structure that holds UMQ ULB receiver information in an extended form.*
- **typedef lbm\_src\_event\_umq\_ulb\_message\_info\_ex\_t\_stct lbm\_src\_event\_umq\_ulb\_message\_info\_ex\_t**  
*Structure that holds UMQ ULB message information in an extended form.*
- **typedef lbm\_ulong\_t(\*) lbm\_str\_hash\_function\_cb (const char \*str)**  
*Application callback for user-supplied topic hash function.*
- **typedef lbm\_ulong\_t(\*) lbm\_str\_hash\_function\_cb\_ex (const char \*str, size\_t strlen, void \*clientd)**  
*Application callback for user-supplied extended version of the topic hash function.*
- **typedef int(\*) lbm\_src\_notify\_function\_cb (const char \*topic\_str, const char \*src\_str, void \*clientd)**  
*Application callback to inform application of the presence of new sources for topics.*
- **typedef int(\*) lbm\_wildcard\_rcv\_compare\_function\_cb (const char \*topic\_str, void \*clientd)**  
*Application callback for application-supplied wildcard matching.*
- **typedef lbm\_uint\_t(\*) lbm\_ume\_rcv\_regid\_function\_cb (const char \*src\_str, lbm\_uint\_t src\_regid, void \*clientd)**  
*Application callback to set the registration ID for a receiver for a specific source.*
- **typedef lbm\_uint\_t(\*) lbm\_ume\_rcv\_regid\_ex\_function\_cb (lbm\_ume\_rcv\_regid\_ex\_func\_info\_t \*info, void \*clientd)**  
*Application callback to set the registration ID for a receiver for a specific source, extended form.*

- **typedef int(\*) lmb\_ume\_src\_force\_reclaim\_function\_cb** (const char \*topic\_str, lmb\_uint\_t seqnum, void \*clientd)  
*Application callback for notification of forced reclamation of retained messages for UMP sources.*
- **typedef int(\*) lmb\_mim\_unrecloss\_function\_cb** (const char \*source\_name, lmb\_uint\_t seqnum, void \*clientd)  
*Application callback in receiving application for notification of unrecoverable lost messages from a multicast immediate message sender.*
- **typedef int(\*) lmb\_ume\_rcv\_recovery\_info\_ex\_function\_cb** (lmb\_ume\_rcv\_recovery\_info\_ex\_func\_info\_t \*info, void \*clientd)  
*Application callback to set the lowest sequence number to be requested during recovery, extended form.*
- **typedef void \*(\* lmb\_rcv\_src\_notification\_create\_function\_cb** (const char \*source\_name, void \*clientd)  
*Application callback for notification of creation of sources for a topic.*
- **typedef int(\*) lmb\_rcv\_src\_notification\_delete\_function\_cb** (const char \*source\_name, void \*clientd, void \*source\_clientd)  
*Application callback for notification of deletion of sources for a topic.*
- **typedef lmb\_str\_hash\_func\_t\_stct lmb\_str\_hash\_func\_t**
- **typedef lmb\_str\_hash\_func\_ex\_t\_stct lmb\_str\_hash\_func\_ex\_t**  
*Structure that holds the hash function callback information.*
- **typedef lmb\_src\_notify\_func\_t\_stct lmb\_src\_notify\_func\_t**  
*Structure that holds the callback for source notifications.*
- **typedef lmb\_wildcard\_rcv\_compare\_func\_t\_stct lmb\_wildcard\_rcv\_compare\_func\_t**  
*Structure that holds the application callback pattern type information for wildcard receivers.*
- **typedef lmb\_ume\_rcv\_regid\_func\_t\_stct lmb\_ume\_rcv\_regid\_func\_t**  
*Structure that holds the application callback for registration ID setting.*
- **typedef lmb\_ume\_rcv\_regid\_ex\_func\_t\_stct lmb\_ume\_rcv\_regid\_ex\_func\_t**  
*Structure that holds the application callback for registration ID setting, extended form.*

- **typedef lbm\_ume\_src\_force\_reclaim\_func\_t\_stct lbm\_ume\_src\_force\_reclaim\_func\_t**  
*Structure that holds the application callback for forced reclamation notifications.*
- **typedef lbm\_mim\_unrecloss\_func\_t\_stct lbm\_mim\_unrecloss\_func\_t**  
*Structure that holds the application callback for multicast immediate message unrecoverable loss notification.*
- **typedef lbm\_ume\_rcv\_recovery\_info\_ex\_func\_t\_stct lbm\_ume\_rcv\_recovery\_info\_ex\_func\_t**  
*Structure that holds the application callback for recovery sequence number information, extended form.*
- **typedef lbm\_ume\_store\_entry\_t\_stct lbm\_ume\_store\_entry\_t**  
*Structure that holds information for a UMP store for configuration purposes.*
- **typedef lbm\_ucast\_resolver\_entry\_t\_stct lbm\_ucast\_resolver\_entry\_t**  
*Structure that holds information for a unicast resolver daemon for configuration purposes.*
- **typedef lbm\_ume\_store\_name\_entry\_t\_stct lbm\_ume\_store\_name\_entry\_t**  
*Structure that holds information for a UMP store by name for configuration purposes.*
- **typedef lbm\_ume\_store\_group\_entry\_t\_stct lbm\_ume\_store\_group\_entry\_t**  
*Structure that holds information for a UMP store group for configuration purposes.*
- **typedef lbm\_rcv\_src\_notification\_func\_t\_stct lbm\_rcv\_src\_notification\_func\_t**  
*Structure that holds the application callback for source status notifications for receivers.*
- **typedef ume\_liveness\_receiving\_context\_t\_stct ume\_liveness\_receiving\_context\_t**  
*Structure that holds the information about a receiving context.*
- **typedef void (\*) lbm\_ume\_ctx\_rcv\_ctx\_notification\_create\_function\_cb (const ume\_liveness\_receiving\_context\_t \*rcv, void \*clientd)**  
*Application callback for notification of detection of a receiving application.*
- **typedef int(\*) lbm\_ume\_ctx\_rcv\_ctx\_notification\_delete\_function\_cb (const ume\_liveness\_receiving\_context\_t \*rcv, void \*clientd, void \*source\_clientd)**  
*Application callback for notification of unresponsiveness of a receiving application.*

- **typedef lbm\_ume\_ctx\_rev\_ctx\_notification\_func\_t\_stct lbm\_ume\_ctx\_rcv\_ctx\_notification\_func\_t**  
*Structure that holds the application callback for receiving context status notifications for source context.*
- **typedef lbm\_umq\_queue\_entry\_t\_stct lbm\_umq\_queue\_entry\_t**  
*Structure that holds information for a UMQ queue registration ID for configuration purposes.*
- **typedef lbm\_umq\_ulb\_receiver\_type\_entry\_t\_stct lbm\_umq\_ulb\_receiver\_type\_entry\_t**  
*Structure that holds information for a UMQ ULB sources receiver type associations with application sets.*
- **typedef lbm\_umq\_ulb\_application\_set\_attr\_t\_stct lbm\_umq\_ulb\_application\_set\_attr\_t**  
*Structure that holds information for a UMQ ULB sources application set attributes.*
- **typedef lbm\_umq\_ulb\_receiver\_type\_attr\_t\_stct lbm\_umq\_ulb\_receiver\_type\_attr\_t**  
*Structure that holds information for a UMQ ULB sources receiver type attributes.*
- **typedef lbm\_context\_t\_stct lbm\_context\_t**
- **typedef int(\*) lbm\_context\_src\_cb\_proc (lbm\_context\_t \*ctx, int event, void \*ed, void \*clientd)**  
*Application context-level callback for events associated with context sources (immediate mode sources and responses).*
- **typedef lbm\_context\_src\_event\_func\_t\_stct lbm\_context\_src\_event\_func\_t**  
*Structure that holds the application callback for context-level source events.*
- **typedef int(\*) lbm\_context\_event\_cb\_proc (lbm\_context\_t \*ctx, int event, void \*ed, void \*clientd)**  
*Application context-level callback for events associated with contexts.*
- **typedef lbm\_context\_event\_func\_t\_stct lbm\_context\_event\_func\_t**  
*Structure that holds the application callback for context-level events.*
- **typedef lbm\_serialized\_response\_t\_stct lbm\_serialized\_response\_t**  
*Structure that holds a serialized UM response object.*
- **typedef lbm\_buff\_t\_stct lbm\_buff\_t**
- **typedef lbm\_wildcard\_rcv\_t\_stct lbm\_wildcard\_rcv\_t**
- **typedef lbm\_hf\_rcv\_t\_stct lbm\_hf\_rcv\_t**

- **typedef lbm\_hfx\_attr\_t\_stct lbm\_hfx\_attr\_t**
- **typedef lbm\_hfx\_t\_stct lbm\_hfx\_t**
- **typedef lbm\_hfx\_recv\_t\_stct lbm\_hfx\_recv\_t**
- **typedef lbm\_topic\_t\_stct lbm\_topic\_t**
- **typedef lbm\_src\_t\_stct lbm\_src\_t**
- **typedef lbm\_recv\_t\_stct lbm\_recv\_t**
- **typedef lbm\_request\_t\_stct lbm\_request\_t**
- **typedef lbm\_response\_t\_stct lbm\_response\_t**
- **typedef lbm\_msg\_fragment\_info\_t\_stct lbm\_msg\_fragment\_info\_t**

*Structure that holds fragment information for UM messages when appropriate.*

- **typedef lbm\_msg\_gateway\_info\_t\_stct lbm\_msg\_gateway\_info\_t**

*Structure that holds originating information for UM messages which arrived via a gateway.*

- **typedef lbm\_msg\_channel\_info\_t\_stct lbm\_msg\_channel\_info\_t**

*Structure that represents UMS Spectrum channel information.*

- **typedef lbm\_ume\_rcv\_ack\_t\_stct lbm\_ume\_rcv\_ack\_t**

- **typedef int(\*) lbm\_immediate\_msg\_cb\_proc (lbm\_context\_t \*ctx, lbm\_msg\_t \*msg, void \*clientd)**

*Application callback for non-topic immediate-mode received messages.*

- **typedef lbm\_context\_rcv\_immediate\_msgs\_func\_t\_stct lbm\_context\_rcv\_immediate\_msgs\_func\_t**

*Structure that holds the application callback for receiving topic-less immediate mode messages.*

- **typedef lbm\_transport\_source\_info\_t\_stct lbm\_transport\_source\_info\_t**

*Structure that holds formatted and parsed transport source strings.*

- **typedef lbm\_uint32\_t(\*) lbm\_src\_cost\_function\_cb (const char \*topic, const lbm\_transport\_source\_info\_t \*transport, lbm\_uint32\_t hop\_count, lbm\_uint32\_t cost, void \*clientd)**

*Application callback to evaluate the cost of a newly discovered source.*

- **typedef lbm\_src\_cost\_func\_t\_stct lbm\_src\_cost\_func\_t**

*Structure that holds the "source\_cost\_evaluation\_function" context attribute.*

- **typedef lbm\_context\_attr\_t\_stct lbm\_context\_attr\_t**

- **typedef lbm\_config\_option\_stct\_t lbm\_config\_option\_t**

- **typedef lbm\_src\_topic\_attr\_t\_stct lbm\_src\_topic\_attr\_t**

- **typedef lbm\_recv\_topic\_attr\_t\_stct lbm\_recv\_topic\_attr\_t**

- **typedef int(\*) lbt\_wildcard\_rcv\_create\_function\_cb** (const char \*topic\_str, lbt\_rcv\_topic\_attr\_t \*attr, void \*clientd)  
*Application callback for wildcard receiver creation.*
- **typedef lbt\_wildcard\_rcv\_create\_func\_t\_stct lbt\_wildcard\_rcv\_create\_func\_t**  
*Structure that holds the receiver creation callback information for wildcard receivers.*
- **typedef int(\*) lbt\_wildcard\_rcv\_delete\_function\_cb** (const char \*topic\_str, void \*clientd)  
*Application callback for wildcard receiver deletion.*
- **typedef lbt\_wildcard\_rcv\_delete\_func\_t\_stct lbt\_wildcard\_rcv\_delete\_func\_t**  
*Structure that holds the receiver deletion callback information for wildcard receivers.*
- **typedef lbt\_wildcard\_rcv\_attr\_t\_stct lbt\_wildcard\_rcv\_attr\_t**
- **typedef lbt\_src\_transport\_stats\_tcp\_t\_stct lbt\_src\_transport\_stats\_tcp\_t**  
*Structure that holds datagram statistics for source TCP transports.*
- **typedef lbt\_src\_transport\_stats\_lbtrm\_t\_stct lbt\_src\_transport\_stats\_lbtrm\_t**  
*Structure that holds datagram statistics for source LBT-RM transports.*
- **typedef lbt\_src\_transport\_stats\_daemon\_t\_stct lbt\_src\_transport\_stats\_daemon\_t**  
*Structure that holds statistics for source daemon mode transport (deprecated).*
- **typedef lbt\_src\_transport\_stats\_lbtru\_t\_stct lbt\_src\_transport\_stats\_lbtru\_t**  
*Structure that holds datagram statistics for source LBT-RU transports.*
- **typedef lbt\_src\_transport\_stats\_lbtipc\_t\_stct lbt\_src\_transport\_stats\_lbtipc\_t**  
*Structure that holds datagram statistics for source LBT-IPC transports.*
- **typedef lbt\_src\_transport\_stats\_lbtrdma\_t\_stct lbt\_src\_transport\_stats\_lbtrdma\_t**  
*Structure that holds datagram statistics for source LBT-RDMA transports.*
- **typedef lbt\_src\_transport\_stats\_t\_stct lbt\_src\_transport\_stats\_t**  
*Structure that holds statistics for source transports.*
- **typedef lbt\_rcv\_transport\_stats\_tcp\_t\_stct lbt\_rcv\_transport\_stats\_tcp\_t**  
*Structure that holds datagram statistics for receiver TCP transports.*

- **typedef lbm\_rcv\_transport\_stats\_lbtrm\_t\_stct lbm\_rcv\_transport\_stats\_lbtrm\_t**  
*Structure that holds datagram statistics for receiver LBT-RM transports.*
- **typedef lbm\_rcv\_transport\_stats\_daemon\_t\_stct lbm\_rcv\_transport\_stats\_daemon\_t**  
*Structure that holds statistics for receiver daemon mode transport (deprecated).*
- **typedef lbm\_rcv\_transport\_stats\_lbtru\_t\_stct lbm\_rcv\_transport\_stats\_lbtru\_t**  
*Structure that holds datagram statistics for receiver LBT-RU transports.*
- **typedef lbm\_rcv\_transport\_stats\_lbtipc\_t\_stct lbm\_rcv\_transport\_stats\_lbtipc\_t**  
*Structure that holds datagram statistics for receiver LBT-IPC transports.*
- **typedef lbm\_rcv\_transport\_stats\_lbtrdma\_t\_stct lbm\_rcv\_transport\_stats\_lbtrdma\_t**  
*Structure that holds datagram statistics for receiver LBT-RDMA transports.*
- **typedef lbm\_rcv\_transport\_stats\_t\_stct lbm\_rcv\_transport\_stats\_t**  
*Structure that holds statistics for receiver transports.*
- **typedef lbm\_event\_queue\_attr\_t\_stct lbm\_event\_queue\_attr\_t**  
• **typedef lbm\_event\_queue\_stats\_t\_stct lbm\_event\_queue\_stats\_t**  
*Structure that holds statistics for an event queue.*
- **typedef lbm\_context\_stats\_t\_stct lbm\_context\_stats\_t**  
*Structure that holds statistics for a context.*
- **typedef int(\*) lbm\_timer\_cb\_proc (lbm\_context\_t \*ctx, const void \*clientd)**  
*Application callback for timer events.*
- **typedef int(\*) lbm\_rcv\_cb\_proc (lbm\_rcv\_t \*rcv, lbm\_msg\_t \*msg, void \*clientd)**  
*Application callback for receiver events.*
- **typedef int(\*) lbm\_fd\_cb\_proc (lbm\_context\_t \*ctx, lbm\_handle\_t handle, lbm\_ulong\_t ev, void \*clientd)**  
*Application callback for events associated with an application file descriptor or socket.*
- **typedef int(\*) lbm\_src\_cb\_proc (lbm\_src\_t \*src, int event, void \*ed, void \*clientd)**  
*Application callback for events associated with a source.*

- **typedef int(\*) lbt\_request\_cb\_proc** (lbt\_request\_t \*req, **lbt\_msg\_t** \*msg, void \*clientd)  
*Application callback for responses returned when a request is sent.*
- **typedef int(\*) lbt\_event\_queue\_monitor\_proc** (lbt\_event\_queue\_t \*evq, int event, size\_t evq\_size, lbt\_ulong\_t event\_delay\_usec, void \*clientd)  
*Application callback for event queue monitor events.*
- **typedef int(\*) lbt\_log\_cb\_proc** (int level, const char \*message, void \*clientd)  
*Application callback for message logging.*
- **typedef int(\*) lbt\_daemon\_event\_cb\_proc** (lbt\_context\_t \*ctx, int event, const char \*info, void \*clientd)  
*Application callback for daemon events.*
- **typedef void(\*) lbt\_event\_queue\_cancel\_cb\_proc** (int dispatch\_thrd, void \*clientd)  
*Application callback for lbt\_\*\_delete\_ex().*
- **typedef lbt\_event\_queue\_cancel\_cb\_info\_t\_stct lbt\_event\_queue\_cancel\_cb\_info\_t**  
*Structure passed to cancel/delete functions so that a cancel callback may be called.*
- **typedef int(\*) lbt\_flight\_size\_set\_inflight\_cb\_proc** (int inflight, void \*clientd)  
*Application callback for lbt\_\*\_flight\_size\_set\_inflight().*
- **typedef void(\*) lbt\_flight\_size\_set\_inflight\_ex\_cb\_proc** (**lbt\_flight\_size\_inflight\_t** \*inflight, void \*clientd)  
*Application callback for lbt\_ume\_flight\_size\_set\_inflight\_ex(). Change the inflight parameter messages and bytes to update the current settings.*
- **typedef lbt\_apphdr\_chain\_iter\_t\_stct lbt\_apphdr\_chain\_iter\_t**
- **typedef lbt\_apphdr\_chain\_elem\_t\_stct lbt\_apphdr\_chain\_elem\_t**  
*Structure that represents an element in an app header chain.*
- **typedef lbt\_msg\_properties\_iter\_t\_stct lbt\_msg\_properties\_iter\_t**  
*A struct used for iterating over properties pointed to by an lbt\_msg\_properties\_t.*
- **typedef lbt\_umq\_msg\_selector\_t\_stct lbt\_umq\_msg\_selector\_t**
- **typedef int(\*) lbt\_cred\_callback\_fn** (const char \*name, size\_t name\_len, const char \*passwd, size\_t passwd\_len, void \*clientd)
- **typedef lbt\_umm\_info\_t\_stct lbt\_umm\_info\_t**  
*Structure for specifying UMM daemon connection options.*

## Enumerations

- enum { **LBM\_OK** = 0, **LBM\_FAILURE** = -1 }

## Functions

- LBMEExpDLL const char \* **lbtm\_version** (void)  
*return the version string compiled into UM.*
- LBMEExpDLL int **lbtm\_context\_dump** (lbtm\_context\_t \*ctx, int \*size, lbtm\_config\_option\_t \*opts)  
*Retrieves all context attribute options.*
- LBMEExpDLL int **lbtm\_context\_attr\_dump** (lbtm\_context\_attr\_t \*cattr, int \*size, lbtm\_config\_option\_t \*opts)  
*Retrieves all context attribute options.*
- LBMEExpDLL int **lbtm\_context\_attr\_option\_size** ()  
*Retrieves the number of options that are of type "context".*
- LBMEExpDLL int **lbtm\_context\_attr\_create** (lbtm\_context\_attr\_t \*\*attr)  
*Create and fill a UM context attribute object with the current default values.*
- LBMEExpDLL int **lbtm\_context\_attr\_create\_default** (lbtm\_context\_attr\_t \*\*attr)  
*Create and fill a UM context attribute object with the initial default values.*
- LBMEExpDLL int **lbtm\_context\_attr\_create\_from\_xml** (lbtm\_context\_attr\_t \*\*attr, const char \*context\_name)  
*Create and fill a UM context attribute object with the current default values for the given context name.*
- LBMEExpDLL int **lbtm\_context\_attr\_set\_from\_xml** (lbtm\_context\_attr\_t \*attr, const char \*context\_name)  
*Fill a UM context attribute object with the current default values for the given context name.*
- LBMEExpDLL int **lbtm\_context\_attr\_delete** (lbtm\_context\_attr\_t \*attr)  
*Delete a UM context attribute object.*
- LBMEExpDLL int **lbtm\_context\_attr\_dup** (lbtm\_context\_attr\_t \*\*attr, const lbtm\_context\_attr\_t \*original)  
*Duplicate a UM context attribute object.*

- LBMDLL int [lbt\\_context\\_attr\\_setopt](#) (lbt\_context\_attr\_t \*attr, const char \*optname, const void \*optval, size\_t optlen)  
*Set an option for the given UM context attribute.*
- LBMDLL int [lbt\\_context\\_attr\\_str\\_setopt](#) (lbt\_context\_attr\_t \*attr, const char \*optname, const char \*optval)  
*Set an option for the given UM context attribute using a string.*
- LBMDLL int [lbt\\_context\\_attr\\_getopt](#) (lbt\_context\_attr\_t \*attr, const char \*optname, void \*optval, size\_t \*optlen)  
*Retrieve the value of an option for the given UM context attribute.*
- LBMDLL int [lbt\\_context\\_attr\\_str\\_getopt](#) (lbt\_context\_attr\_t \*attr, const char \*optname, char \*optval, size\_t \*optlen)  
*Retrieve the textual value of an option for the given UM context attribute.*
- LBMDLL int [lbt\\_context\\_create](#) (lbt\_context\_t \*\*ctxp, const lbt\_context\_attr\_t \*attr, [lbt\\_daemon\\_event\\_cb\\_proc](#) proc, void \*clientd)  
*Create and initialize an lbt\_context\_t object.*
- LBMDLL int [lbt\\_context\\_reactor\\_only\\_create](#) (lbt\_context\_t \*\*ctxp, const lbt\_context\_attr\_t \*attr)  
*Create and initialize an lbt\_context\_t object suitable for FD and timers only.*
- LBMDLL int [lbt\\_context\\_delete](#) (lbt\_context\_t \*ctx)  
*Delete a UM context object.*
- LBMDLL int [lbt\\_context\\_delete\\_ex](#) (lbt\_context\_t \*ctx, [lbt\\_event\\_queue\\_cancel\\_cb\\_info\\_t](#) \*cbinfo)  
*Delete a UM context object with an application callback indicating when the context is fully deleted. This extended version of the context delete function requires the configuration option queue\_cancellation\_callbacks\_enabled be set to 1.*
- LBMDLL int [lbt\\_context\\_topic\\_resolution\\_request](#) (lbt\_context\_t \*ctx, lbt\_ushort\_t flags, lbt\_ulong\_t interval\_msec, lbt\_ulong\_t duration\_sec)  
*Request Topic Advertisements (sources), Topic Queries (receivers), and/or Wildcard Topic Queries (wildcard receivers) in the configured topic resolution address domain. Since Advertisements and Queries can become quiescent after a period defined by the Topic Resolution configuration attributes, this function will schedule Topic Resolution Requests at the given interval and duration. Contexts that receive these requests will respond with one advertisement per source and/or one query per receiver as appropriate. These requests will be ignored for topics that are not quiescent. Note that requests are only sent on the outgoing address and are only received on the incoming address. Responses to the request will similarly be sent only on the outgoing address.*

- LBMEExpDLL int [lbt\\_context\\_setopt](#) (lbt\_context\_t \*ctx, const char \*optname, const void \*optval, size\_t optlen)
 

*Set an option value within the given ctx.*
- LBMEExpDLL int [lbt\\_context\\_str\\_setopt](#) (lbt\_context\_t \*ctx, const char \*optname, const char \*optval)
 

*Set an option value within the given ctx.*
- LBMEExpDLL int [lbt\\_context\\_getopt](#) (lbt\_context\_t \*ctx, const char \*optname, void \*optval, size\_t \*optlen)
 

*Retrieve an option value within the given ctx.*
- LBMEExpDLL int [lbt\\_context\\_str\\_getopt](#) (lbt\_context\_t \*ctx, const char \*optname, char \*optval, size\_t \*optlen)
 

*Retrieve the textual option value within the given ctx.*
- LBMEExpDLL int [lbt\\_context\\_rcv\\_immediate\\_msgs](#) (lbt\_context\_t \*ctx, [lbt\\_immediate\\_msg\\_cb\\_proc](#) proc, void \*clientd, lbt\_event\_queue\_t \*evq)
 

*Set the callback procedure and delivery method for non-topic immediate messages.*
- LBMEExpDLL int [lbt\\_context\\_rcv\\_immediate\\_topic\\_msgs](#) (lbt\_context\_t \*ctx, [lbt\\_immediate\\_msg\\_cb\\_proc](#) proc, void \*clientd, lbt\_event\_queue\_t \*evq)
 

*Set the callback procedure and delivery method for immediate messages to a topic for which there is no receiver.*
- LBMEExpDLL int [lbt\\_context\\_set\\_name](#) (lbt\_context\_t \*ctx, const char \*name)
 

*Set the name associated with a context.*
- LBMEExpDLL int [lbt\\_context\\_get\\_name](#) (lbt\_context\_t \*ctx, char \*name, size\_t \*size)
 

*Get the name associated with a context.*
- LBMEExpDLL int [lbt\\_license\\_file](#) (const char \*licfile)
 

*Initialize the UM license from the contents of a disk file. This function will only be effective if it is called before any other UM API function.*
- LBMEExpDLL int [lbt\\_license\\_str](#) (const char \*licstr)
 

*Initialize the UM license from a string. This function will only be effective if it is called before any other UM API function.*
- LBMEExpDLL int [lbt\\_config](#) (const char \*fname)

*Set one or more options from the contents of a disk file. This function will only be effective if it is called before any other UM API function.*

- LBMEExpDLL int **lbt\_config\_xml\_file** (const char \*url, const char \*application\_name)  
*Load a UM XML configuration file.*
- LBMEExpDLL int **lbt\_config\_xml\_string** (const char \*xml\_data, const char \*application\_name)  
*Load UM XML configuration data.*
- LBMEExpDLL int **lbt\_log** (**lbt\_log\_cb\_proc** proc, void \*clientd)  
*Set a callback function to be called for UM log messages (warnings, notices, etc.).*
- LBMEExpDLL void **lbt\_logf** (int level, const char \*format,...)  
*Log a message. This is an entry to the UM logging mechanism.*
- LBMEExpDLL const char \* **lbt\_errmsg** (void)  
*Return an ASCII string containing the error message last encountered by this thread.*
- LBMEExpDLL int **lbt\_errnum** (void)  
*Return the error number last encountered by this thread.*
- LBMEExpDLL int **lbt\_win32\_static\_thread\_attach** (void)  
*Instructs UM that a new thread will be calling UM functions.*
- LBMEExpDLL int **lbt\_win32\_static\_thread\_detach** (void)  
*Instructs UM that a new thread is done calling UM functions.*
- LBMEExpDLL int **lbt\_schedule\_timer** (**lbt\_context\_t** \*ctx, **lbt\_timer\_cb\_proc** proc, void \*clientd, **lbt\_event\_queue\_t** \*evq, **lbt\_ulong\_t** delay)  
*Schedule a timer that calls proc when it expires.*
- LBMEExpDLL int **lbt\_schedule\_timer\_recurring** (**lbt\_context\_t** \*ctx, **lbt\_timer\_cb\_proc** proc, void \*clientd, **lbt\_event\_queue\_t** \*evq, **lbt\_ulong\_t** delay)  
*Schedule a recurring timer that calls proc when it expires.*
- LBMEExpDLL int **lbt\_cancel\_timer** (**lbt\_context\_t** \*ctx, int id, void \*\*clientdp)  
*Cancel a previously scheduled timer identified by id.*
- LBMEExpDLL int **lbt\_cancel\_timer\_ex** (**lbt\_context\_t** \*ctx, int id, void \*\*clientdp, **lbt\_event\_queue\_cancel\_cb\_info\_t** \*cbinfo)

*Extended cancel a previously scheduled timer identified by id.*

- LBMEExpDLL int [lbtipc\\_process\\_events](#) (lbtipc\_context\_t \*ctx, lbtipc\_ulong\_t msec)

*Process internal events in the given UM context object.*

- LBMEExpDLL int [lbtipc\\_unblock](#) (lbtipc\_context\_t \*ctx)

*Unblock a sequential mode UM context.*

- LBMEExpDLL int [lbtipc\\_process\\_lbtipc\\_messages](#) (lbtipc\_context\_t \*ctx, lbtipc\_ulong\_t msec, lbtipc\_ulong\_t loop\_count)

*Process LBT-IPC messages received.*

- LBMEExpDLL int [lbtipc\\_unregister\\_fd](#) (lbtipc\_context\_t \*ctx)

*Unblock a sequential mode LBT-IPC processing loop.*

- LBMEExpDLL int [lbtipc\\_register\\_fd](#) (lbtipc\_context\_t \*ctx, lbtipc\_handle\_t handle, lbtipc\_fd\_cb\_proc proc, void \*clientd, lbtipc\_event\_queue\_t \*evq, lbtipc\_ulong\_t ev)

*Register a file descriptor/socket for events that calls proc when a given event occurs.*

- LBMEExpDLL int [lbtipc\\_cancel\\_fd](#) (lbtipc\_context\_t \*ctx, lbtipc\_handle\_t handle, lbtipc\_ulong\_t ev)

*Cancel a previously registered file descriptor/socket event.*

- LBMEExpDLL int [lbtipc\\_cancel\\_fd\\_ex](#) (lbtipc\_context\_t \*ctx, lbtipc\_handle\_t handle, lbtipc\_ulong\_t ev, [lbtipc\\_event\\_queue\\_cancel\\_cb\\_info\\_t](#) \*cbinfo)

*Extended cancel a previously registered file descriptor/socket event.*

- LBMEExpDLL int [lbtipc\\_src\\_topic\\_dump](#) (lbtipc\_src\_t \*src, int \*size, lbtipc\_config\_option\_t \*opts)

*Retrieves all source topic attribute options.*

- LBMEExpDLL int [lbtipc\\_src\\_topic\\_attr\\_dump](#) (lbtipc\_src\_topic\_attr\_t \*sattr, int \*size, lbtipc\_config\_option\_t \*opts)

*Retrieves all source topic attribute options.*

- LBMEExpDLL int [lbtipc\\_src\\_topic\\_attr\\_option\\_size](#) ()

*Retrieves the number of options that are of type "topic".*

- LBMEExpDLL int [lbtipc\\_src\\_topic\\_alloc](#) (lbtipc\_src\_topic\_t \*\*topic, lbtipc\_context\_t \*ctx, const char \*symbol, const lbtipc\_src\_topic\_attr\_t \*attr)

*Turn a Topic string into a UM topic object usable by sources.*

- LBMDLL int **lbtm\_src\_topic\_attr\_create** (lbtm\_src\_topic\_attr\_t \*\*attr)  
*Create and fill a UM source topic attribute object with the current default values.*
- LBMDLL int **lbtm\_src\_topic\_attr\_create\_default** (lbtm\_src\_topic\_attr\_t \*\*attr)  
*Create and fill a UM source topic attribute object with the initial default values.*
- LBMDLL int **lbtm\_src\_topic\_attr\_create\_from\_xml** (lbtm\_src\_topic\_attr\_t \*\*attr, const char \*context\_name, const char \*topicname)  
*Create and fill a UM source topic attribute object with the current default values for the given topic name.*
- LBMDLL int **lbtm\_src\_topic\_attr\_set\_from\_xml** (lbtm\_src\_topic\_attr\_t \*attr, const char \*context\_name, const char \*topicname)  
*Fill a UM source topic attribute object with the current default values for the given topic name.*
- LBMDLL int **lbtm\_src\_topic\_attr\_delete** (lbtm\_src\_topic\_attr\_t \*attr)  
*Delete a source topic attribute object.*
- LBMDLL int **lbtm\_src\_topic\_attr\_dup** (lbtm\_src\_topic\_attr\_t \*\*attr, const lbtm\_src\_topic\_attr\_t \*original)  
*Duplicate a UM source topic attribute object.*
- LBMDLL int **lbtm\_src\_topic\_attr\_setopt** (lbtm\_src\_topic\_attr\_t \*attr, const char \*optname, const void \*optval, size\_t optlen)  
*Set an option value within the given source topic attribute.*
- LBMDLL int **lbtm\_src\_topic\_attr\_str\_setopt** (lbtm\_src\_topic\_attr\_t \*attr, const char \*optname, const char \*optval)  
*Set an option value within the given source topic attribute.*
- LBMDLL int **lbtm\_src\_topic\_attr\_getopt** (lbtm\_src\_topic\_attr\_t \*attr, const char \*optname, void \*optval, size\_t optlen)  
*Retrieve an option value within the given source topic attribute.*
- LBMDLL int **lbtm\_src\_topic\_attr\_str\_getopt** (lbtm\_src\_topic\_attr\_t \*attr, const char \*optname, char \*optval, size\_t \*optlen)  
*Retrieve a textual option value within the given source topic attribute.*
- LBMDLL int **lbtm\_rcv\_topic\_lookup** (lbtm\_topic\_t \*\*topicp, lbtm\_context\_t \*ctx, const char \*symbol, const lbtm\_rcv\_topic\_attr\_t \*attr)

*Turn a Topic string into a UM topic object usable by receivers.*

- LBMEExpDLL int [lbt\\_rcv\\_topic\\_dump](#) (lbt\_rcv\_t \*rcv, int \*size, lbt\_config\_option\_t \*opts)  
*Retrieves all receiver topic attribute options.*
- LBMEExpDLL int [lbt\\_rcv\\_topic\\_attr\\_dump](#) (lbt\_rcv\_topic\_attr\_t \*rattr, int \*size, lbt\_config\_option\_t \*opts)  
*Retrieves all receiver topic attribute options.*
- LBMEExpDLL int [lbt\\_rcv\\_topic\\_attr\\_option\\_size](#) ()  
*Retrieves the number of options that are of type "source topic".*
- LBMEExpDLL int [lbt\\_rcv\\_topic\\_attr\\_create](#) (lbt\_rcv\_topic\_attr\_t \*\*attr)  
*Create and fill a UM receiver topic attribute object with the current default values.*
- LBMEExpDLL int [lbt\\_rcv\\_topic\\_attr\\_create\\_default](#) (lbt\_rcv\_topic\_attr\_t \*\*attr)  
*Create and fill a UM receiver topic attribute object with the initial default values.*
- LBMEExpDLL int [lbt\\_rcv\\_topic\\_attr\\_create\\_from\\_xml](#) (lbt\_rcv\_topic\_attr\_t \*\*attr, const char \*context\_name, const char \*topicname)  
*Create and fill a UM receiver topic attribute object with the current default values for the given topic name.*
- LBMEExpDLL int [lbt\\_rcv\\_topic\\_attr\\_set\\_from\\_xml](#) (lbt\_rcv\_topic\_attr\_t \*attr, const char \*context\_name, const char \*topicname)  
*Fill a UM receiver topic attribute object with the current default values for the given topic name.*
- LBMEExpDLL int [lbt\\_rcv\\_topic\\_attr\\_delete](#) (lbt\_rcv\_topic\_attr\_t \*attr)  
*Delete a receiver topic attribute object.*
- LBMEExpDLL int [lbt\\_rcv\\_topic\\_attr\\_dup](#) (lbt\_rcv\_topic\_attr\_t \*\*attr, const lbt\_rcv\_topic\_attr\_t \*original)  
*Duplicate a UM receiver topic attribute object.*
- LBMEExpDLL int [lbt\\_rcv\\_topic\\_attr\\_setopt](#) (lbt\_rcv\_topic\_attr\_t \*attr, const char \*optname, const void \*optval, size\_t optlen)  
*Set an option value within the given receiver topic attribute.*
- LBMEExpDLL int [lbt\\_rcv\\_topic\\_attr\\_str\\_setopt](#) (lbt\_rcv\_topic\_attr\_t \*attr, const char \*optname, const char \*optval)

*Set an option value within the given receiver topic attribute.*

- LBMEExpDLL int **lbt\_rcv\_topic\_attr\_setopt** (lbt\_rcv\_topic\_attr\_t \*attr, const char \*optname, void \*optval, size\_t \*optlen)

*Retrieve an option value within the given receiver topic attribute.*

- LBMEExpDLL int **lbt\_rcv\_topic\_attr\_str\_getopt** (lbt\_rcv\_topic\_attr\_t \*attr, const char \*optname, char \*optval, size\_t \*optlen)

*Retrieve a textual option value within the given receiver topic attribute.*

- LBMEExpDLL int **lbt\_src\_channel\_create** (lbt\_src\_channel\_info\_t \*\*chnp, lbt\_src\_t \*src, lbt\_uint32\_t channel\_num)

*Create a channel info object to send messages with the given channel\_num.*

- LBMEExpDLL int **lbt\_src\_channel\_delete** (lbt\_src\_channel\_info\_t \*chn)

*Release the resources associated with a source channel.*

- LBMEExpDLL int **lbt\_src\_create** (lbt\_src\_t \*\*srcp, lbt\_context\_t \*ctx, lbt\_topic\_t \*topic, **lbt\_src\_cb\_proc** proc, void \*clientd, lbt\_event\_queue\_t \*evq)

*Create a UM source that will send messages to the given topic.*

- LBMEExpDLL int **lbt\_event\_queue\_dump** (lbt\_event\_queue\_t \*evq, int \*size, lbt\_config\_option\_t \*opts)

*Retrieves all event queue attribute options.*

- LBMEExpDLL int **lbt\_event\_queue\_attr\_dump** (lbt\_event\_queue\_attr\_t \*eatr, int \*size, lbt\_config\_option\_t \*opts)

*Retrieves all event queue attribute options.*

- LBMEExpDLL int **lbt\_event\_queue\_attr\_option\_size** ()

*Retrieves the number of options that are of type "event queue".*

- LBMEExpDLL int **lbt\_rcv\_create** (lbt\_rcv\_t \*\*rcvp, lbt\_context\_t \*ctx, lbt\_topic\_t \*topic, **lbt\_rcv\_cb\_proc** proc, void \*clientd, lbt\_event\_queue\_t \*evq)

*Create a UM receiver that will receive messages sent to the given topic.*

- LBMEExpDLL int **lbt\_rcv\_subscribe\_channel** (lbt\_rcv\_t \*rcv, lbt\_uint32\_t channel, **lbt\_rcv\_cb\_proc** proc, void \*clientd)

*Subscribe to a channel, with an optional callback and clientd data pointer.*

- LBMEExpDLL int **lbt\_rcv\_unsubscribe\_channel** (lbt\_rcv\_t \*rcv, lbt\_uint32\_t channel)

*Discontinue an existing channel subscription.*

- LBMEExpDLL int [lmb\\_rcv\\_unsubscribe\\_channel\\_ex](#) (lmb\_rcv\_t \*rcv, lmb\_uint32\_t channel, [lmb\\_event\\_queue\\_cancel\\_cb\\_info\\_t](#) \*cbinfo)

*Discontinue an existing channel subscription with an application callback indicating when all messages on the channel have been delivered. This extended version of the unsubscribe function requires the configuration option queue\_cancellation\_callbacks\_enabled be set to 1.*

- LBMEExpDLL int [lmb\\_wildcard\\_rcv\\_subscribe\\_channel](#) (lmb\_wildcard\_rcv\_t \*wrcv, lmb\_uint32\_t channel, [lmb\\_rcv\\_cb\\_proc](#) proc, void \*clientd)

*Subscribe to a channel, with an optional callback and clientd data pointer.*

- LBMEExpDLL int [lmb\\_wildcard\\_rcv\\_unsubscribe\\_channel](#) (lmb\_wildcard\_rcv\_t \*wrcv, lmb\_uint32\_t channel)

*Discontinue an existing channel subscription.*

- LBMEExpDLL int [lmb\\_wildcard\\_rcv\\_unsubscribe\\_channel\\_ex](#) (lmb\_wildcard\_rcv\_t \*wrcv, lmb\_uint32\_t channel, [lmb\\_event\\_queue\\_cancel\\_cb\\_info\\_t](#) \*cbinfo)

*Discontinue an existing channel subscription with an application callback indicating when all messages on the channel have been delivered. This extended version of the unsubscribe function requires the configuration option queue\_cancellation\_callbacks\_enabled be set to 1.*

- LBMEExpDLL int [lmb\\_src\\_delete](#) (lmb\_src\_t \*src)

*Delete a UM source object.*

- LBMEExpDLL int [lmb\\_src\\_delete\\_ex](#) (lmb\_src\_t \*src, [lmb\\_event\\_queue\\_cancel\\_cb\\_info\\_t](#) \*cbinfo)

*Extended delete a UM source object.*

- LBMEExpDLL lmb\_context\_t \* [lmb\\_context\\_from\\_src](#) (lmb\_src\_t \*src)

*Retrieve the UM context object associated with a UM source object.*

- LBMEExpDLL lmb\_topic\_t \* [lmb\\_topic\\_from\\_src](#) (lmb\_src\_t \*src)

*Retrieve the UM topic object associated with a UM source object.*

- LBMEExpDLL lmb\_event\_queue\_t \* [lmb\\_event\\_queue\\_from\\_src](#) (lmb\_src\_t \*src)

*Retrieve the UM event queue object associated with a UM source object.*

- LBMEExpDLL int [lmb\\_rcv\\_delete](#) (lmb\_rcv\_t \*rcv)

*Delete a UM receiver object.*

- LBMEExpDLL int [lbt\\_rcv\\_delete\\_ex](#) (lbt\_rcv\_t \*rcv, lbt\_event\_queue\_cancel\_cb\_info\_t \*cbinfo)  
*Extended delete a UM receiver object.*
- LBMEExpDLL lbt\_context\_t \* [lbt\\_context\\_from\\_rcv](#) (lbt\_rcv\_t \*rcv)  
*Retrieve the UM context object associated with a UM receiver object.*
- LBMEExpDLL lbt\_event\_queue\_t \* [lbt\\_event\\_queue\\_from\\_rcv](#) (lbt\_rcv\_t \*rcv)  
*Retrieve the UM event queue object associated with a UM receiver object.*
- LBMEExpDLL int [lbt\\_src\\_setopt](#) (lbt\_src\_t \*src, const char \*optname, const void \*optval, size\_t optlen)  
*Set an option value within the given src.*
- LBMEExpDLL int [lbt\\_src\\_str\\_setopt](#) (lbt\_src\_t \*src, const char \*optname, const char \*optval)  
*Set an option value within the given src.*
- LBMEExpDLL int [lbt\\_src\\_getopt](#) (lbt\_src\_t \*src, const char \*optname, void \*optval, size\_t \*optlen)  
*Retrieve an option value within the given src.*
- LBMEExpDLL int [lbt\\_src\\_str\\_getopt](#) (lbt\_src\_t \*src, const char \*optname, char \*optval, size\_t \*optlen)  
*Retrieve a textual option value within the given src.*
- LBMEExpDLL int [lbt\\_rcv\\_setopt](#) (lbt\_rcv\_t \*rcv, const char \*optname, const void \*optval, size\_t optlen)  
*Set an option value within the given rcv.*
- LBMEExpDLL int [lbt\\_rcv\\_str\\_setopt](#) (lbt\_rcv\_t \*rcv, const char \*optname, const char \*optval)  
*Set an option value within the given rcv.*
- LBMEExpDLL int [lbt\\_rcv\\_getopt](#) (lbt\_rcv\_t \*rcv, const char \*optname, void \*optval, size\_t \*optlen)  
*Retrieve an option value within the given rcv.*
- LBMEExpDLL int [lbt\\_rcv\\_str\\_getopt](#) (lbt\_rcv\_t \*rcv, const char \*optname, char \*optval, size\_t \*optlen)  
*Retrieve a textual option value within the given rcv.*

- LBMEExpDLL int [lbt\\_src\\_send](#) (lbt\_src\_t \*src, const char \*msg, size\_t len, int flags)  
*Send a message to the topic associated with a UM source.*
- LBMEExpDLL int [lbt\\_src\\_send\\_ex](#) (lbt\_src\_t \*src, const char \*msg, size\_t len, int flags, [lbt\\_src\\_send\\_ex\\_info\\_t](#) \*info)  
*Extended send of a message to the topic associated with a UM source.*
- LBMEExpDLL int [lbt\\_src\\_flush](#) (lbt\_src\_t \*src)  
*Send messages from both the explicit and implicit batches ASAP.*
- LBMEExpDLL int [lbt\\_src\\_sendv](#) (lbt\_src\_t \*src, const [lbt\\_iovec\\_t](#) \*iov, int num, int flags)  
*Send a set of messages to the topic associated with a UM source.*
- LBMEExpDLL int [lbt\\_src\\_sendv\\_ex](#) (lbt\_src\_t \*src, const [lbt\\_iovec\\_t](#) \*iov, int num, int flags, [lbt\\_src\\_send\\_ex\\_info\\_t](#) \*info)  
*Extended send of a set of messages to the topic associated with a UM source.*
- LBMEExpDLL int [lbt\\_rev\\_msg\\_source\\_clientd](#) (lbt\_rev\_t \*recv, const char \*source, void \*source\_clientd)  
*Set the pointer value to set in the messages received for the given receiver from a specific source.*
- LBMEExpDLL int [lbt\\_src\\_retrieve\\_transport\\_stats](#) (lbt\_src\_t \*src, [lbt\\_src\\_transport\\_stats\\_t](#) \*stats)  
*Retrieve the transport statistics for the transport used by the given source.*
- LBMEExpDLL int [lbt\\_src\\_reset\\_transport\\_stats](#) (lbt\_src\_t \*src)  
*Reset the transport statistics for the transport used by the given source.*
- LBMEExpDLL int [lbt\\_rev\\_retrieve\\_transport\\_stats](#) (lbt\_rev\_t \*recv, const char \*source, [lbt\\_rev\\_transport\\_stats\\_t](#) \*stats)  
*Retrieve the transport statistics for the transport used by the given receiver from a specific source.*
- LBMEExpDLL int [lbt\\_rev\\_reset\\_transport\\_stats](#) (lbt\_rev\_t \*recv, const char \*source)  
*Reset the transport statistics for the transport used by the given receiver from a specific source.*
- LBMEExpDLL int [lbt\\_rev\\_retrieve\\_all\\_transport\\_stats](#) (lbt\_rev\_t \*recv, int \*num, [lbt\\_rev\\_transport\\_stats\\_t](#) \*stats)

*Retrieve the transport stats for all the sources seen by the given receiver.*

- LBMEExpDLL int **lbtm\_rcv\_retrieve\_all\_transport\_stats\_ex** (lbtm\_rcv\_t \*rcv, int \*num, int size, **lbtm\_rcv\_transport\_stats\_t** \*stats)
- LBMEExpDLL int **lbtm\_rcv\_reset\_all\_transport\_stats** (lbtm\_rcv\_t \*rcv)

*Reset the transport stats for all the sources seen by the given receiver.*

- LBMEExpDLL int **lbtm\_context\_retrieve\_rcv\_transport\_stats** (lbtm\_context\_t \*ctx, int \*num, **lbtm\_rcv\_transport\_stats\_t** \*stats)

*Retrieve the transport stats for all receivers in a given context.*

- LBMEExpDLL int **lbtm\_context\_retrieve\_rcv\_transport\_stats\_ex** (lbtm\_context\_t \*ctx, int \*num, int size, **lbtm\_rcv\_transport\_stats\_t** \*stats)
- LBMEExpDLL int **lbtm\_context\_reset\_rcv\_transport\_stats** (lbtm\_context\_t \*ctx)

*Reset the transport stats for all receivers in a given context.*

- LBMEExpDLL int **lbtm\_context\_retrieve\_src\_transport\_stats** (lbtm\_context\_t \*ctx, int \*num, **lbtm\_src\_transport\_stats\_t** \*stats)

*Retrieve the transport stats for all the sources in a given context.*

- LBMEExpDLL int **lbtm\_context\_retrieve\_src\_transport\_stats\_ex** (lbtm\_context\_t \*ctx, int \*num, int size, **lbtm\_src\_transport\_stats\_t** \*stats)
- LBMEExpDLL int **lbtm\_context\_reset\_src\_transport\_stats** (lbtm\_context\_t \*ctx)

*Reset the transport stats for all the sources in a given context.*

- LBMEExpDLL int **lbtm\_event\_queue\_retrieve\_stats** (lbtm\_event\_queue\_t \*evq, **lbtm\_event\_queue\_stats\_t** \*stats)

*Retrieve the stats for an event queue.*

- LBMEExpDLL int **lbtm\_event\_queue\_reset\_stats** (lbtm\_event\_queue\_t \*evq)

*Reset the stats for an event queue.*

- LBMEExpDLL int **lbtm\_context\_retrieve\_stats** (lbtm\_context\_t \*ctx, **lbtm\_context\_stats\_t** \*stats)

*Retrieve the stats for a context.*

- LBMEExpDLL int **lbtm\_context\_reset\_stats** (lbtm\_context\_t \*ctx)

*Reset the stats for a context.*

- LBMEExpDLL int **lbtm\_context\_retrieve\_im\_src\_transport\_stats** (lbtm\_context\_t \*ctx, int \*num, int size, **lbtm\_src\_transport\_stats\_t** \*stats)

*Retrieve the IM source stats for a context.*

- LBMEExpDLL int `lbt_context_reset_im_src_transport_stats` (`lbt_context_t *ctx`)
 

*Reset the IM source stats for a context.*
- LBMEExpDLL int `lbt_context_retrieve_im_rcv_transport_stats` (`lbt_context_t *ctx, int *num, int size, lbt_rcv_transport_stats_t *stats`)
 

*Retrieve the IM receiver stats for a context.*
- LBMEExpDLL int `lbt_context_reset_im_rcv_transport_stats` (`lbt_context_t *ctx`)
 

*Reset the IM receiver stats for a context.*
- LBMEExpDLL int `lbt_msg_retain` (`lbt_msg_t *msg`)
 

*Instruct UM that the API is going to retain ownership of a UM message object.*
- LBMEExpDLL int `lbt_msg_is_fragment` (`lbt_msg_t *msg`)
 

*Retrieve fragment information from a UM message.*
- LBMEExpDLL int `lbt_msg_retrieve_fragment_info` (`lbt_msg_t *msg, lbt_msg_fragment_info_t *info`)
 

*Returns 1 if lbt message is a fragment, else 0 is returned.*
- LBMEExpDLL int `lbt_msg_retrieve_gateway_info` (`lbt_msg_t *msg, lbt_msg_gateway_info_t *info`)
 

*Retrieve gateway information from a UM message.*
- LBMEExpDLL int `lbt_msg_retrievemsgid` (`lbt_msg_t *msg, lbt_umq_msgid_t *id`)
 

*Retrieve UMQ Message ID information from a UM message.*
- LBMEExpDLL int `lbt_msg_retrieve_umq_index` (`lbt_msg_t *msg, lbt_umq_index_info_t *info`)
 

*Retrieve UMQ index information from a UM message.*
- LBMEExpDLL int `lbt_msg_retrieve_delivery_latency` (`lbt_msg_t *msg, lbt_int64_t *latency_nsecs`)
- LBMEExpDLL int `lbt_msg_delete` (`lbt_msg_t *msg`)
 

*Delete a UM message object.*
- LBMEExpDLL int `lbt_msg_ume_send_explicit_ack` (`lbt_msg_t *msg`)
 

*Send an Explicit UMP ACK for a UM message object.*
- LBMEExpDLL int `lbt_msg_ume_can_send_explicit_ack` (`lbt_msg_t *msg`)

*Check to see if Explicit UMP ACK for a UM message object can be called.*

- LBMEExpDLL int [lbt\\_src\\_ume\\_deregister](#) (lbt\_src\_t \*src)  
*Deregister a source from the UMP stores.*
- LBMEExpDLL int [lbt\\_rcv\\_ume\\_deregister](#) (lbt\_rcv\_t \*rcv)  
*Deregister a receiver from all known UMP stores.*
- LBMEExpDLL int [lbt\\_wildcard\\_rcv\\_ume\\_deregister](#) (lbt\_wildcard\_rcv\_t \*wrcv)  
*Deregister a wildcard receiver from all known UMP stores.*
- LBMEExpDLL int [lbt\\_msg\\_umq\\_reassign](#) (lbt\_msg\_t \*msg, int flags)  
*Do not acknowledge the given message and instead request that the message be reassigned.*
- LBMEExpDLL int [lbt\\_rcv\\_umq\\_deregister](#) (lbt\_rcv\_t \*rcv, const char \*queue\_name)  
*De-Register the given receiver from the given UMQ queue or all UMQ queues.*
- LBMEExpDLL int [lbt\\_rcv\\_umq\\_index\\_stop\\_assignment](#) (lbt\_rcv\_t \*rcv, const char \*queue\_name)  
*Stop assignment of new UMQ indices to the given receiver from the given UMQ queue or all UMQ queues.*
- LBMEExpDLL int [lbt\\_rcv\\_umq\\_index\\_start\\_assignment](#) (lbt\_rcv\_t \*rcv, const char \*queue\_name)  
*Start assignment of new UMQ indices to the given receiver from the given UMQ queue or all UMQ queues.*
- LBMEExpDLL int [lbt\\_rcv\\_umq\\_index\\_reserve](#) (lbt\_rcv\_t \*rcv, const char \*queue\_name, lbt\_umq\_index\_info\_t \*index\_info)  
*Instruct the given UMQ queue(s) to reserve an index for assignment to this receiver.*
- LBMEExpDLL int [lbt\\_rcv\\_umq\\_index\\_release](#) (lbt\_rcv\_t \*rcv, const char \*queue\_name, lbt\_umq\_index\_info\_t \*index\_info)  
*Instruct the given UMQ queue(s) to release the given UMQ index that is assigned to the given receiver.*
- LBMEExpDLL int [lbt\\_wildcard\\_rcv\\_umq\\_index\\_stop\\_assignment](#) (lbt\_wildcard\_rcv\_t \*wrcv, const char \*queue\_name)  
*Stop assignment of new UMQ indices to the given wildcard receiver from the given UMQ queue or all UMQ queues.*

- LBMEExpDLL int [lbt\\_wildcard\\_rcv\\_umq\\_index\\_start\\_assignment](#) (lbt\_wildcard\_rcv\_t \*wrcv, const char \*queue\_name)
 

*Start assignment of new UMQ indices to the given wildcard receiver from the given UMQ queue or all UMQ queues.*
- LBMEExpDLL int [lbt\\_wildcard\\_rcv\\_umq\\_index\\_release](#) (lbt\_wildcard\_rcv\_t \*wrcv, const char \*queue\_name, [lbt\\_umq\\_index\\_info\\_t](#) \*index\_info)
 

*Instruct the given UMQ queue(s) to release the given UMQ index that is assigned to the given wildcard receiver.*
- LBMEExpDLL int [lbt\\_wildcard\\_rcv\\_umq\\_deregister](#) (lbt\_wildcard\_rcv\_t \*wrcv, const char \*queue\_name)
 

*De-Register the given wildcard receiver from the given UMQ queue or all UMQ queues.*
- LBMEExpDLL int [lbt\\_send\\_response](#) (lbt\_response\_t \*resp, const char \*data, size\_t len, int flags)
 

*Send a response for a given resp response.*
- LBMEExpDLL int [lbt\\_response\\_delete](#) (lbt\_response\_t \*resp)
 

*Delete a UM response object.*
- LBMEExpDLL int [lbt\\_serialized\\_response\\_delete](#) (lbt\_serialized\_response\_t \*serialized\_response)
 

*Delete a UM serialized response object.*
- LBMEExpDLL lbt\_serialized\_response\_t \* [lbt\\_serialize\\_response](#) (lbt\_response\_t \*resp)
 

*Serialize a UM response object.*
- LBMEExpDLL lbt\_response\_t \* [lbt\\_deserialize\\_response](#) (lbt\_context\_t \*ctx, lbt\_serialized\_response\_t \*serialized\_response)
 

*De-serialize a UM response object.*
- LBMEExpDLL int [lbt\\_send\\_request](#) (lbt\_request\_t \*\*reqp, lbt\_src\_t \*src, const char \*data, size\_t len, [lbt\\_request\\_cb\\_proc](#) proc, void \*clientd, lbt\_event\_queue\_t \*evq, int send\_flags)
 

*Send a request on the given src that contains the given data.*
- LBMEExpDLL int [lbt\\_send\\_request\\_ex](#) (lbt\_request\_t \*\*reqp, lbt\_src\_t \*src, const char \*data, size\_t len, [lbt\\_request\\_cb\\_proc](#) proc, void \*clientd, lbt\_event\_queue\_t \*evq, int send\_flags, [lbt\\_src\\_send\\_ex\\_info\\_t](#) \*exinfo)
 

*Send a request on the given src that contains the given data.*

- LBMEExpDLL int **lbtm\_request\_delete** (lbtm\_request\_t \*req)  
*Delete a UM request object.*
- LBMEExpDLL int **lbtm\_request\_delete\_ex** (lbtm\_request\_t \*req, **lbtm\_event\_queue\_cancel\_cb\_info\_t** \*cbinfo)  
*Extended delete a UM request object.*
- LBMEExpDLL int **lbtm\_event\_queue\_create** (lbtm\_event\_queue\_t \*\*evqp, **lbtm\_event\_queue\_monitor\_proc** proc, void \*clientd, const lbtm\_event\_queue\_attr\_t \*attr)  
*Create a UM event queue object.*
- LBMEExpDLL int **lbtm\_event\_queue\_attr\_create** (lbtm\_event\_queue\_attr\_t \*\*attr)  
*Create and fill a UM event queue attribute object with the current default values.*
- LBMEExpDLL int **lbtm\_event\_queue\_attr\_create\_default** (lbtm\_event\_queue\_attr\_t \*\*attr)  
*Create and fill a UM event queue attribute object with the initial default values.*
- LBMEExpDLL int **lbtm\_event\_queue\_attr\_create\_from\_xml** (lbtm\_event\_queue\_attr\_t \*\*attr, const char \*event\_queue\_name)  
*Create and fill a UM event queue attribute object with the current default values for the given event queue name.*
- LBMEExpDLL int **lbtm\_event\_queue\_attr\_set\_from\_xml** (lbtm\_event\_queue\_attr\_t \*attr, const char \*event\_queue\_name)  
*Fill a UM event queue attribute object with the current default values for the given event queue name.*
- LBMEExpDLL int **lbtm\_event\_queue\_attr\_delete** (lbtm\_event\_queue\_attr\_t \*attr)  
*Delete an event queue attribute object.*
- LBMEExpDLL int **lbtm\_event\_queue\_attr\_dup** (lbtm\_event\_queue\_attr\_t \*\*attr, const lbtm\_event\_queue\_attr\_t \*original)  
*Duplicate a UM event queue attribute object.*
- LBMEExpDLL int **lbtm\_event\_queue\_attr\_setopt** (lbtm\_event\_queue\_attr\_t \*attr, const char \*optname, const void \*optval, size\_t optlen)  
*Set an option value within the given event queue attribute.*
- LBMEExpDLL int **lbtm\_event\_queue\_attr\_str\_setopt** (lbtm\_event\_queue\_attr\_t \*attr, const char \*optname, const char \*optval)

*Set an option value within the given event queue attribute.*

- LBMEExpDLL int [lbt\\_event\\_queue\\_attr\\_getopt](#) (lbt\_event\_queue\_attr\_t \*attr, const char \*optname, void \*optval, size\_t \*optlen)

*Retrieve an option value within the given event queue attribute.*

- LBMEExpDLL int [lbt\\_event\\_queue\\_attr\\_str\\_getopt](#) (lbt\_event\_queue\_attr\_t \*attr, const char \*optname, char \*optval, size\_t \*optlen)

*Retrieve a textual option value within the given event queue attribute.*

- LBMEExpDLL int [lbt\\_event\\_queue\\_setopt](#) (lbt\_event\_queue\_t \*evq, const char \*optname, const void \*optval, size\_t optlen)

*Set an option value within the given evq.*

- LBMEExpDLL int [lbt\\_event\\_queue\\_str\\_setopt](#) (lbt\_event\_queue\_t \*evq, const char \*optname, const char \*optval)

*Set an option value within the given event queue.*

- LBMEExpDLL int [lbt\\_event\\_queue\\_getopt](#) (lbt\_event\_queue\_t \*evq, const char \*optname, void \*optval, size\_t \*optlen)

*Retrieve an option value within the given event queue.*

- LBMEExpDLL int [lbt\\_event\\_queue\\_str\\_getopt](#) (lbt\_event\_queue\_t \*evq, const char \*optname, char \*optval, size\_t \*optlen)

*Retrieve a textual option value within the given event queue.*

- LBMEExpDLL int [lbt\\_event\\_dispatch](#) (lbt\_event\_queue\_t \*evq, lbt\_ulong\_t tmo)

*Dispatch waiting events to appropriate callback functions.*

- LBMEExpDLL int [lbt\\_event\\_dispatch\\_unblock](#) (lbt\_event\_queue\_t \*evq)

*Unblock the given UM event queue object so that a thread waiting in lbt\_event\_dispatch returns as soon as feasible.*

- LBMEExpDLL int [lbt\\_event\\_queue\\_size](#) (lbt\_event\_queue\_t \*evq)

*Determine the number of queued events in the event queue.*

- LBMEExpDLL int [lbt\\_event\\_queue\\_shutdown](#) (lbt\_event\_queue\_t \*evq)

*Shutdown the event queue by purging any pending events and not allowing additional events to be added to the queue.*

- LBMEExpDLL int [lbt\\_event\\_queue\\_delete](#) (lbt\_event\_queue\_t \*evq)

*Delete a given UM event queue object.*

- LBMEExpDLL int [lbtm\\_unicast\\_immediate\\_message](#) (lbtm\_context\_t \*ctx, const char \*target, const char \*topic, const char \*data, size\_t len, int flags)  
*Unicast an immediate message to the target and topic. Note that immediate messages are processed somewhat less efficiently than source-based messages.*
- LBMEExpDLL int [lbtm\\_unicast\\_immediate\\_request](#) (lbtm\_request\_t \*\*reqp, lbtm\_context\_t \*ctx, const char \*target, const char \*topic, const char \*data, size\_t len, [lbtm\\_request\\_cb\\_proc](#) proc, void \*clientd, lbtm\_event\_queue\_t \*evq, int flags)  
*Unicast an immediate request to the target and topic. Note that immediate messages are processed somewhat less efficiently than source-based messages.*
- LBMEExpDLL int [lbtm\\_queue\\_immediate\\_message](#) (lbtm\_context\_t \*ctx, const char \*qname, const char \*topic, const char \*data, size\_t len, int flags, [lbtm\\_src\\_send\\_ex\\_info\\_t](#) \*info)  
*Submit a message to a given UMQ queue and to a given topic. Note that immediate messages are processed somewhat less efficiently than source-based messages.*
- LBMEExpDLL int [lbtm\\_multicast\\_immediate\\_message](#) (lbtm\_context\_t \*ctx, const char \*topic, const char \*data, size\_t len, int flags)  
*Multicast an immediate message to the topic. Note that immediate messages are processed somewhat less efficiently than source-based messages.*
- LBMEExpDLL int [lbtm\\_multicast\\_immediate\\_request](#) (lbtm\_request\_t \*\*reqp, lbtm\_context\_t \*ctx, const char \*topic, const char \*data, size\_t len, [lbtm\\_request\\_cb\\_proc](#) proc, void \*clientd, lbtm\_event\_queue\_t \*evq, int flags)  
*Multicast an immediate request to the target and topic. Note that immediate messages are processed somewhat less efficiently than source-based messages.*
- LBMEExpDLL int [lbtm\\_wildcard\\_rcv\\_dump](#) (lbtm\_wildcard\_rcv\_t \*wrcv, int \*size, lbtm\_config\_option\_t \*opts)  
*Retrieves all wildcard receiver attribute options.*
- LBMEExpDLL int [lbtm\\_wildcard\\_rcv\\_attr\\_dump](#) (lbtm\_wildcard\_rcv\_attr\_t \*wattr, int \*size, lbtm\_config\_option\_t \*opts)  
*Retrieves all wildcard receiver attribute options.*
- LBMEExpDLL int [lbtm\\_wildcard\\_rcv\\_attr\\_option\\_size](#) ()  
*Retrieves the number of options that are of type "wildcard receiver".*
- LBMEExpDLL int [lbtm\\_wildcard\\_rcv\\_attr\\_create](#) (lbtm\_wildcard\_rcv\_attr\_t \*\*attr)  
*Create and fill a UM wildcard receiver attribute object with the current default values.*

- LBMEExpDLL int [lbt\\_wildcard\\_rcv\\_attr\\_create\\_default](#) (lbt\_wildcard\_rcv\_attr\_t \*\*attr)  
*Create and fill a UM wildcard receiver attribute object with the initial default values.*
- LBMEExpDLL int [lbt\\_wildcard\\_rcv\\_attr\\_create\\_from\\_xml](#) (lbt\_wildcard\_rcv\_attr\_t \*\*attr, const char \*context\_name, const char \*pattern, int pattern\_type)  
*Create and fill a UM wildcard receiver attribute object with the current default values for the given topic name.*
- LBMEExpDLL int [lbt\\_wildcard\\_rcv\\_attr\\_set\\_from\\_xml](#) (lbt\_wildcard\_rcv\_attr\_t \*attr, const char \*context\_name, const char \*pattern, int pattern\_type)  
*Fill a UM wildcard receiver attribute object with the current default values for the given topic name.*
- LBMEExpDLL int [lbt\\_wildcard\\_rcv\\_attr\\_delete](#) (lbt\_wildcard\_rcv\_attr\_t \*attr)  
*Delete a wildcard receiver attribute object.*
- LBMEExpDLL int [lbt\\_wildcard\\_rcv\\_attr\\_dup](#) (lbt\_wildcard\_rcv\_attr\_t \*\*attr, const lbt\_wildcard\_rcv\_attr\_t \*original)  
*Duplicate a UM wildcard receiver attribute object.*
- LBMEExpDLL int [lbt\\_wildcard\\_rcv\\_attr\\_setopt](#) (lbt\_wildcard\_rcv\_attr\_t \*attr, const char \*optname, const void \*optval, size\_t optlen)  
*Set an option value within the given wildcard receiver attribute.*
- LBMEExpDLL int [lbt\\_wildcard\\_rcv\\_attr\\_str\\_setopt](#) (lbt\_wildcard\_rcv\_attr\_t \*attr, const char \*optname, const char \*optval)  
*Set an option value within the given wildcard receiver attribute.*
- LBMEExpDLL int [lbt\\_wildcard\\_rcv\\_attr\\_getopt](#) (lbt\_wildcard\_rcv\_attr\_t \*attr, const char \*optname, void \*optval, size\_t optlen)  
*Retrieve an option value within the given wildcard receiver attribute.*
- LBMEExpDLL int [lbt\\_wildcard\\_rcv\\_attr\\_str\\_getopt](#) (lbt\_wildcard\_rcv\_attr\_t \*attr, const char \*optname, char \*optval, size\_t optlen)  
*Retrieve a textual option value within the given wildcard receiver attribute.*
- LBMEExpDLL int [lbt\\_wildcard\\_rcv\\_setopt](#) (lbt\_wildcard\_rcv\_t \*wrcv, const char \*optname, const void \*optval, size\_t optlen)  
*Set an option value within the given wrcv.*

- LBMDLL int **lbt\_wildcard\_rcv\_str\_setopt** (lbt\_wildcard\_rcv\_t \*wrcv, const char \*optname, const char \*optval)  
*Set an option value within the given wrcv.*
- LBMDLL int **lbt\_wildcard\_rcv\_getopt** (lbt\_wildcard\_rcv\_t \*wrcv, const char \*optname, void \*optval, size\_t \*optlen)  
*Retrieve an option value within the given wrcv.*
- LBMDLL int **lbt\_wildcard\_rcv\_str\_getopt** (lbt\_wildcard\_rcv\_t \*wrcv, const char \*optname, char \*optval, size\_t \*optlen)  
*Retrieve the textual option value within the given wrcv.*
- LBMDLL int **lbt\_wildcard\_rcv\_create** (lbt\_wildcard\_rcv\_t \*\*wrcvp, lbt\_context\_t \*ctx, const char \*pattern, const lbt\_rcv\_topic\_attr\_t \*tattr, const lbt\_wildcard\_rcv\_attr\_t \*wattr, **lbt\_rcv\_cb\_proc** proc, void \*clientd, lbt\_event\_queue\_t \*evq)  
*Create a UM wildcard receiver that will receive messages sent to any topic matching the given pattern. Note that if wildcard queries are enabled, LBM will query a maximum of 250 patterns (receivers).*
- LBMDLL int **lbt\_wildcard\_rcv\_delete** (lbt\_wildcard\_rcv\_t \*wrcv)  
*Delete a UM wildcard receiver object.*
- LBMDLL int **lbt\_wildcard\_rcv\_delete\_ex** (lbt\_wildcard\_rcv\_t \*wrcv, lbt\_event\_queue\_cancel\_cb\_info\_t \*cbinfo)  
*Extended delete a UM wildcard receiver object.*
- LBMDLL lbt\_context\_t \* **lbt\_context\_from\_wildcard\_rcv** (lbt\_wildcard\_rcv\_t \*wrcv)  
*Retrieve the LBM context object associated with a UM wildcard receiver object.*
- LBMDLL lbt\_event\_queue\_t \* **lbt\_event\_queue\_from\_wildcard\_rcv** (lbt\_wildcard\_rcv\_t \*wrcv)  
*Retrieve the LBM event queue object associated with a UM wildcard receiver object.*
- LBMDLL int **lbt\_hf\_src\_create** (lbt\_src\_t \*\*srcp, lbt\_context\_t \*ctx, lbt\_topic\_t \*topic, **lbt\_src\_cb\_proc** proc, void \*clientd, lbt\_event\_queue\_t \*evq)  
*Create a UM Hot Failover (HF) source that will send messages to the given topic. See <https://communities.informatica.com/infakb/faq/5/Pages/80060.aspx> for details and restrictions.*
- LBMDLL int **lbt\_hf\_src\_send** (lbt\_src\_t \*src, const char \*msg, size\_t len, lbt\_uint\_t sqn, int flags)

*Send a Hot Failover (HF) message to the topic associated with a UM source. See <https://communities.informatica.com/infakb/faq/5/Pages/80060.aspx> for details and restrictions.*

- LBMEExpDLL int `lbt_hf_src_send_ex` (`lbt_src_t *src`, `const char *msg`, `size_t len`, `lbt_uint_t sqn`, `int flags`, `lbt_src_send_ex_info_t *exinfo`)  
*Send a Hot Failover (HF) message to the topic associated with a UM source. See <https://communities.informatica.com/infakb/faq/5/Pages/80060.aspx> for details and restrictions.*
- LBMEExpDLL int `lbt_hf_src_sendv` (`lbt_src_t *src`, `const lbt_ovec_t *iov`, `int num`, `lbt_uint_t sqn`, `int flags`)  
*Send a set of Hot Failover (HF) messages to the topic associated with a UM source. See <https://communities.informatica.com/infakb/faq/5/Pages/80060.aspx> for details and restrictions.*
- LBMEExpDLL int `lbt_hf_rcv_topic_dump` (`lbt_hf_rcv_t *hfrcv`, `int *size`, `lbt_config_option_t *opts`)  
*Retrieves all receiver attribute options for an HF receiver.*
- LBMEExpDLL int `lbt_hf_src_sendv_ex` (`lbt_src_t *src`, `const lbt_ovec_t *iov`, `int num`, `lbt_uint_t sqn`, `int flags`, `lbt_src_send_ex_info_t *exinfo`)  
*Extended send of a set of Hot Failover (HF) messages to the topic associated with a UM source.*
- LBMEExpDLL int `lbt_hf_src_send_rcv_reset` (`lbt_src_t *src`, `int flags`, `lbt_src_send_ex_info_t *exinfo`)  
*Send a message that will reset order and loss information for hot failover receivers on this topic.*
- LBMEExpDLL int `lbt_hf_rcv_create` (`lbt_hf_rcv_t **hfrcvp`, `lbt_context_t *ctx`, `lbt_topic_t *topic`, `lbt_rcv_cb_proc proc`, `void *clientd`, `lbt_event_queue_t *evq`)  
*Create and LBM receiver that will receive LBM Hot Failover (HF) messages sent to the given topic. See <https://communities.informatica.com/infakb/faq/5/Pages/80060.aspx> for details and restrictions.*
- LBMEExpDLL int `lbt_hf_rcv_delete` (`lbt_hf_rcv_t *hfrcv`)  
*Delete a UM Hot Failover (HF) receiver object. See <https://communities.informatica.com/infakb/faq/5/Pages/80060.aspx> for details and restrictions.*
- LBMEExpDLL int `lbt_hf_rcv_delete_ex` (`lbt_hf_rcv_t *hfrcv`, `lbt_event_queue_cancel_cb_info_t *cbinfo`)

*Extended delete a UM Hot Failover (HF) receiver object. See <https://communities.informatica.com/infakb/faq/5/Pages/80060.aspx> for details and restrictions.*

- LBMEExpDLL `lbt_hf_rcv_t * lbt_hf_rcv_from_rcv` (`lbt_rcv_t *rcv`)  
*Return the LBM Hot Failover (HF) receiver object (if any) from a UM receiver object.*
- LBMEExpDLL `lbt_rcv_t * lbt_rcv_from_hf_rcv` (`lbt_hf_rcv_t *hfrcv`)  
*Return the LBM receiver object associated with a UM Hot Failover (HF) receiver object.*
- LBMEExpDLL int `lbt_hfx_dump` (`lbt_hfx_t *hfx, int *size, lbt_config_option_t *opts`)  
*Retrieves all HFX attribute options.*
- LBMEExpDLL int `lbt_hfx_attr_dump` (`lbt_hfx_attr_t *attr, int *size, lbt_config_option_t *opts`)  
*Retrieves all HFX attribute options.*
- LBMEExpDLL int `lbt_hfx_attr_option_size` ()  
*Retrieves the number of options that are of type "hfx". The function returns the number of entries that are of type "hfx".*
- LBMEExpDLL int `lbt_hfx_attr_create` (`lbt_hfx_attr_t **attr`)  
*Create and fill a UM HFX attribute object with the current default values.*
- LBMEExpDLL int `lbt_hfx_attr_create_default` (`lbt_hfx_attr_t **attr`)  
*Create and fill a UM HFX attribute object with the initial default values.*
- LBMEExpDLL int `lbt_hfx_attr_create_from_xml` (`lbt_hfx_attr_t **attr, const char *topicname`)  
*Create and fill a UM hfx attribute object with the current default values for the given topic name.*
- LBMEExpDLL int `lbt_hfx_attr_set_from_xml` (`lbt_hfx_attr_t *attr, const char *topicname`)  
*Fill a UM hfx attribute object with the current default values for the given topic name.*
- LBMEExpDLL int `lbt_hfx_attr_delete` (`lbt_hfx_attr_t *attr`)  
*Delete a UM hfx attribute object.*
- LBMEExpDLL int `lbt_hfx_attr_dup` (`lbt_hfx_attr_t **attr, const lbt_hfx_attr_t *original`)  
*Duplicate a UM hfx attribute object.*

- LBMDLL int [lbf\\_hfx\\_attr\\_setopt](#) (lbf\_hfx\_attr\_t \*attr, const char \*optname, const void \*optval, size\_t optlen)  
*Set an option for the given LBM hfx attribute.*
- LBMDLL int [lbf\\_hfx\\_attr\\_str\\_setopt](#) (lbf\_hfx\_attr\_t \*attr, const char \*optname, const char \*optval)  
*Set an option for the given LBM hfx attribute using a string.*
- LBMDLL int [lbf\\_hfx\\_attr\\_getopt](#) (lbf\_hfx\_attr\_t \*attr, const char \*optname, void \*optval, size\_t \*optlen)  
*Retrieve the value of an option for the given LBM hfx attribute.*
- LBMDLL int [lbf\\_hfx\\_attr\\_str\\_getopt](#) (lbf\_hfx\_attr\_t \*attr, const char \*optname, char \*optval, size\_t \*optlen)  
*Retrieve the textual value of an option for the given LBM hfx attribute.*
- LBMDLL int [lbf\\_hfx\\_setopt](#) (lbf\_hfx\_t \*hfx, const char \*optname, const void \*optval, size\_t optlen)  
*Set an option value within the given hfx.*
- LBMDLL int [lbf\\_hfx\\_str\\_setopt](#) (lbf\_hfx\_t \*hfx, const char \*optname, const char \*optval)  
*Set an option value within the given hfx.*
- LBMDLL int [lbf\\_hfx\\_getopt](#) (lbf\_hfx\_t \*hfx, const char \*optname, void \*optval, size\_t \*optlen)  
*Retrieve an option value within the given hfx.*
- LBMDLL int [lbf\\_hfx\\_str\\_getopt](#) (lbf\_hfx\_t \*hfx, const char \*optname, char \*optval, size\_t \*optlen)  
*Retrieve the textual option value within the given hfx.*
- LBMDLL int [lbf\\_hfx\\_create](#) (lbf\_hfx\_t \*\*hfxp, lbf\_hfx\_attr\_t \*cattr, const char \*symbol, [lbf\\_rcv\\_cb\\_proc](#) proc, lbf\_event\_queue\_t \*evq)  
*Create and initialize an lbf\_hfx\_t object.*
- LBMDLL int [lbf\\_hfx\\_delete](#) (lbf\_hfx\_t \*hfx)  
*Delete a UM hfx object.*
- LBMDLL int [lbf\\_hfx\\_delete\\_ex](#) (lbf\_hfx\_t \*hfx, [lbf\\_event\\_queue\\_cancel\\_cb\\_info\\_t](#) \*cbinfo)

*Delete an LBM hfx object and receive a callback when the deletion is complete. Delete an LBM HFX object, with an application callback indicating when the object is fully cancelled. This extended version of the delete function requires the configuration option queue\_cancellation\_callbacks\_enabled to be set to 1 if an event queue is in use. Unlike.*

- LBMEExpDLL int [lbg\\_hfx\\_rcv\\_topic\\_dump](#) (lbg\_hfx\_rcv\_t \*hfxrcv, int \*size, lbg\_config\_option\_t \*opts)

*Retrieves all receiver attribute options for an HFX receiver.*

- LBMEExpDLL int [lbg\\_hfx\\_rcv\\_create](#) (lbg\_hfx\_rcv\_t \*\*hfrcvp, lbg\_hfx\_t \*hfx, lbg\_context\_t \*ctx, lbg\_rcv\_topic\_attr\_t \*rattr, void \*clientd)

*Create a HFX receiver.*

- LBMEExpDLL lbg\_rcv\_t \* [lbg\\_rcv\\_from\\_hfx\\_rcv](#) (lbg\_hfx\_rcv\_t \*hfxrcv)

*Retrieve the underlying receiver from an lbg\_hfx\_rcv\_t.*

- LBMEExpDLL int [lbg\\_hfx\\_rcv\\_delete](#) (lbg\_hfx\_rcv\_t \*hfrcv)

*Delete a HFX receiver.*

- LBMEExpDLL int [lbg\\_hfx\\_rcv\\_delete\\_ex](#) (lbg\_hfx\_rcv\_t \*hfrcv, lbg\_event\_queue\_cancel\_cb\_info\_t \*cbinfo)

*Extended delete a UM HFX receiver object.*

- LBMEExpDLL void [lbg\\_debug\\_filename](#) (const char \*filename)

*Set the file to receive LBM debug log entries.*

- LBMEExpDLL void [lbg\\_debug\\_mask](#) (lbg\_uint64\_t mask)

*Set the debug mask for LBM debug log entries.*

- LBMEExpDLL void [lbg\\_debug\\_noflush](#) (int on)

*Set the noflush flag for the debug logs. This can dramatically increase performance when a debug mask is set.*

- LBMEExpDLL int [lbg\\_debug\\_dump](#) (const char \*filename, int append)

*Dump a running rollback debug log to the given filename.*

- LBMEExpDLL void [lbg\\_set\\_lbtrm\\_loss\\_rate](#) (int rate)

*Dynamically set the LBT-RM loss rate.*

- LBMEExpDLL void [lbg\\_set\\_lbtrm\\_src\\_loss\\_rate](#) (int rate)

*Dynamically set the LBT-RM source loss rate.*

- LBMEExpDLL void [lbg\\_set\\_lbtru\\_loss\\_rate](#) (int rate)

*Dynamically set the LBT-RU loss rate.*

- LBMEExpDLL void [lbt\\_set\\_lbtr\\_src\\_loss\\_rate](#) (int rate)

*Dynamically set the LBT-RU source loss rate.*

- LBMEExpDLL int [lbt\\_transport\\_source\\_parse](#) (const char \*source, [lbt\\_transport\\_source\\_info\\_t](#) \*info, size\_t infosize)

*Parse a UM transport source string into its components.*

- LBMEExpDLL int [lbt\\_transport\\_source\\_format](#) (const [lbt\\_transport\\_source\\_info\\_t](#) \*info, size\_t infosize, char \*source, size\_t \*size)

*Format a UM transport source string from its components.*

- LBMEExpDLL int [lbt\\_apphdr\\_chain\\_create](#) ([lbt\\_apphdr\\_chain\\_t](#) \*\*chain)

*Create a new app header chain that can be used to include metadata with a message.*

- LBMEExpDLL int [lbt\\_apphdr\\_chain\\_delete](#) ([lbt\\_apphdr\\_chain\\_t](#) \*chain)

*Delete an app header chain previously created with [lbt\\_apphdr\\_chain\\_create](#).*

- LBMEExpDLL int [lbt\\_apphdr\\_chain\\_append\\_elem](#) ([lbt\\_apphdr\\_chain\\_t](#) \*chain, [lbt\\_apphdr\\_chain\\_elem\\_t](#) \*elem)

*Appends a user-created app header to an app header chain.*

- LBMEExpDLL int [lbt\\_apphdr\\_chain\\_iter\\_create](#) ([lbt\\_apphdr\\_chain\\_iter\\_t](#) \*\*chain\_iter, [lbt\\_apphdr\\_chain\\_t](#) \*chain)

*Create an iterator (an [lbt\\_apphdr\\_chain\\_iter\\_t](#) structure) to point to the first element in an apphdr chain.*

- LBMEExpDLL int [lbt\\_apphdr\\_chain\\_iter\\_create\\_from\\_msg](#) ([lbt\\_apphdr\\_chain\\_iter\\_t](#) \*\*chain\_iter, [lbt\\_msg\\_t](#) \*msg)

*Create an iterator (an [lbt\\_apphdr\\_chain\\_iter\\_t](#) structure) to point to the first element in an apphdr chain associated with a UM message.*

- LBMEExpDLL int [lbt\\_apphdr\\_chain\\_iter\\_delete](#) ([lbt\\_apphdr\\_chain\\_iter\\_t](#) \*chain\_iter)

*Delete an iterator allocated by one of the [lbt\\_apphdr\\_chain\\_iter](#) functions.*

- LBMEExpDLL int [lbt\\_apphdr\\_chain\\_iter\\_first](#) ([lbt\\_apphdr\\_chain\\_iter\\_t](#) \*\*chain\_iter)

*Initializes an app header chain iterator to the first element in the chain.*

- LBMEExpDLL int [lbt\\_apphdr\\_chain\\_iter\\_done](#) ([lbt\\_apphdr\\_chain\\_iter\\_t](#) \*\*chain\_iter)

*Tests an `lbt_apphdr_chain_iter_t` iterator to see if more elements in the chain remain.*

- LBMEExpDLL int `lbt_apphdr_chain_iter_next` (`lbt_apphdr_chain_iter_t` \*\*`chain_iter`)

*Advances the iterator to the next element in an app header chain, if any.*

- LBMEExpDLL `lbt_apphdr_chain_elem_t` \* `lbt_apphdr_chain_iter_current` (`lbt_apphdr_chain_iter_t` \*\*`chain_iter`)

*Returns the current element of an app header chain pointed to by an `lbt_apphdr_chain_iter_t` iterator.*

- LBMEExpDLL int `lbt_msg_properties_create` (`lbt_msg_properties_t` \*\*`properties`)

*Creates a new properties object, used for sending messages with properties.*

- LBMEExpDLL int `lbt_msg_properties_delete` (`lbt_msg_properties_t` \*`properties`)

*Deletes a properties object.*

- LBMEExpDLL int `lbt_msg_properties_set` (`lbt_msg_properties_t` \*`properties`, const char \*`name`, const void \*`value`, int `type`, size\_t `size`)

*Sets the value of the property with the specified name. Each property name may be associated with one and only one value.*

- LBMEExpDLL int `lbt_msg_properties_clear` (`lbt_msg_properties_t` \*`properties`, const char \*`name`)

*Clear the value associated with the name in the specified properties object.*

- LBMEExpDLL int `lbt_msg_properties_get` (`lbt_msg_properties_t` \*`properties`, const char \*`name`, void \*`value`, int \*`type`, size\_t \*`size`)

*Gets the value of the property with the specified name.*

- LBMEExpDLL int `lbt_msg_properties_iter_create` (`lbt_msg_properties_iter_t` \*\*`iterp`)

*Creates a new msg properties iterator. The newly created iterator is not associated with any properties object. Use.*

- LBMEExpDLL int `lbt_msg_properties_iter_delete` (`lbt_msg_properties_iter_t` \*`iter`)

*Deletes an `lbt_msg_properties_iterator`.*

- LBMEExpDLL int `lbt_msg_properties_iter_first` (`lbt_msg_properties_iter_t` \*`iter`, `lbt_msg_properties_t` \*`properties`)

*Begin iterating over an.*

- LBMEExpDLL int [lbt\\_msg\\_properties\\_iter\\_next](#) (lbt\_msg\_properties\_iter\_t \*iter)  
*Iterate to the next property in an.*
- LBMEExpDLL int [lbt\\_src\\_get\\_inflight](#) (lbt\_src\_t \*src, int type, int \*inflight, [lbt\\_flight\\_size\\_set\\_inflight\\_cb\\_proc](#) proc, void \*clientd)  
*Retrieves the current number of inflight messages of a given type from the src pointed to by lbt\_src\_t \*src.*
- LBMEExpDLL int [lbt\\_src\\_get\\_inflight\\_ex](#) (lbt\_src\_t \*src, int type, [lbt\\_flight\\_size\\_inflight\\_t](#) \*inflight, [lbt\\_flight\\_size\\_set\\_inflight\\_ex\\_cb\\_proc](#) proc, void \*clientd)  
*Retrieves the current number of inflight information of a given type from the src pointed to by lbt\_src\_t \*src.*
- LBMEExpDLL int [lbt\\_ctx\\_umq\\_get\\_inflight](#) (lbt\_context\_t \*ctx, const char \*qname, int \*inflight, [lbt\\_flight\\_size\\_set\\_inflight\\_cb\\_proc](#) proc, void \*clientd)  
*Retrieves the current number of inflight UMQ messages from the ctx pointed to by lbt\_context\_t \*ctx.*
- LBMEExpDLL int [lbt\\_ume\\_src\\_msg\\_stable](#) (lbt\_src\_t \*src, lbt\_uint32\_t sqn)  
*Mark a specific sqn as stable at a store, triggering a source event notification if configured to do so. Also adjusts the current number of inflight messages for the src if necessary.*
- LBMEExpDLL int [lbt\\_umq\\_ctx\\_msg\\_stable](#) (lbt\_context\_t \*ctx, const char \*qname, [lbt\\_umqmsgid\\_t](#) \*msg\_id)  
*Mark a specific msg\_id as stable at qname, triggering a source event notification if configured to do so. Also adjusts the current number of inflight messages for the src if necessary.*
- LBMEExpDLL lbt\_ume\_rcv\_ack\_t \* [lbt\\_msg\\_extract\\_ume\\_ack](#) (lbt\_msg\_t \*msg)  
*Retrieves the ack structure from a UMP message.*
- LBMEExpDLL int [lbt\\_ume\\_ack\\_delete](#) (lbt\_ume\_rcv\_ack\_t \*ack)  
*Deletes an ack structure.*
- LBMEExpDLL int [lbt\\_ume\\_ack\\_send\\_explicit\\_ack](#) (lbt\_ume\_rcv\_ack\_t \*ack, lbt\_uint\_t sqn)  
*Sends an explicit ack up to the sequence number provided.*

- LBMDLL int [lbt\\_ctx\\_umq\\_queue\\_topic\\_list](#) (lbt\_context\_t \*ctx, const char \*queue\_name, [lbt\\_async\\_operation\\_func\\_t](#) \*async\_opfunc)  
*Retrieves a list of currently available topics from a queue (asynchronous operation).*
- LBMDLL int [lbt\\_umq\\_msg\\_selector\\_create](#) (lbt\_umq\_msg\_selector\_t \*\*selector, char \*str, lbt\_uint16\_t len)  
*Create an umq message selector (an lbt\_umq\_msg\_selector\_t structure).*
- LBMDLL int [lbt\\_umq\\_msg\\_selector\\_delete](#) (lbt\_umq\_msg\_selector\_t \*selector)  
*Delete the lbt\_umq\_msg\_selector\_t object previously created with lbt\_umq\_msg\_selector\_create.*
- LBMDLL int [lbt\\_rcv\\_umq\\_queue\\_msg\\_list](#) (lbt\_rcv\_t \*rcv, const char \*queue\_name, lbt\_umq\_msg\_selector\_t \*selector, [lbt\\_async\\_operation\\_func\\_t](#) \*async\_opfunc)  
*Retrieves a list of all currently-queued messages from a queue (asynchronous operation).*
- LBMDLL int [lbt\\_rcv\\_umq\\_queue\\_msg\\_retrieve](#) (lbt\_rcv\_t \*rcv, const char \*queue\_name, [lbt\\_umqmsgid\\_t](#) \*msgids, int num\_msgids, [lbt\\_async\\_operation\\_func\\_t](#) \*async\_opfunc)  
*Retrieves a set of queued messages from the queue (asynchronous operation).*
- LBMDLL int [lbt\\_async\\_operation\\_status](#) (lbt\_async\_operation\_handle\_t handle, int flags)  
*Query the current status of an outstanding asynchronous operation.*
- LBMDLL int [lbt\\_async\\_operation\\_cancel](#) (lbt\_async\_operation\_handle\_t handle, int flags)  
*Cancel an outstanding asynchronous operation.*
- LBMDLL int [lbt\\_auth\\_set\\_credentials](#) (lbt\_context\_t \*ctx, const char \*name, size\_t name\_len, const char \*passwd, size\_t passwd\_len, lbt\_cred\_callback\_fn cbfn, void \*clientd, int auth\_required)  
*Set the user's credential and authentication requirement.*
- LBMDLL int [lbt\\_authstorage\\_open\\_storage\\_xml](#) (char \*filename)  
*Create the storage object from XML password file.*
- LBMDLL void [lbt\\_authstorage\\_close\\_storage\\_xml](#) (void)  
*Release the storage object.*

- LBMEExpDLL int [lbt\\_authstorage\\_checkpermission](#) (char \*username, char \*command)  
*Check if the user is authorized to execute the specified command.*
- LBMEExpDLL int [lbt\\_authstorage\\_addtpnam](#) (const char \*username, const char \*pass, unsigned char flags)  
*Add the new user credential to the password file.*
- LBMEExpDLL int [lbt\\_authstorage\\_deltptnam](#) (const char \*username)  
*Delete the user credential from the password file.*
- LBMEExpDLL int [lbt\\_authstorage\\_user\\_add\\_role](#) (const char \*username, const char \*role)  
*Add one role entry for the user to the password file.*
- LBMEExpDLL int [lbt\\_authstorage\\_user\\_del\\_role](#) (const char \*username, const char \*role)  
*Delete the role entry for the user from the password file.*
- LBMEExpDLL int [lbt\\_authstorage\\_load\\_roletable](#) ()  
*Load the role table from the password file.*
- LBMEExpDLL int [lbt\\_authstorage\\_unload\\_roletable](#) ()  
*Unload the role table.*
- LBMEExpDLL int [lbt\\_authstorage\\_roletable\\_add\\_role\\_action](#) (const char \*rolename, const char \*action)  
*Add a new authorized action for the specified role.*
- LBMEExpDLL int [lbt\\_authstorage\\_print\\_roletable](#) ()  
*Print the role table saved in the internal data object.*
- LBMEExpDLL int [lbt\\_set\\_umm\\_info](#) ([lbt\\_umm\\_info\\_t](#) \*info)  
*Connect to and retrieve configuration from a UMM daemon.*
- LBMEExpDLL int [lbt\\_is\\_ume\\_capable](#) (void)  
*Determine if the LBM library is capable of UME operations.*
- LBMEExpDLL int [lbt\\_is\\_umq\\_capable](#) (void)  
*Determine if the LBM library is capable of UMQ operations.*
- LBMEExpDLL void [lbt\\_seterr](#) (int eno, const char \*str)
- LBMEExpDLL void [lbt\\_seterrf](#) (int eno, const char \*format,...)

- LBMEExpDLL const char \* **lbt\_strerror** (void)
- LBMEExpDLL const char \* **lbt\_strerror\_errnum** (int errnum)
- LBMEExpDLL void **lbt\_sock\_init** ()
- LBMEExpDLL lbt\_uint64\_t **lbt\_create\_random\_id** ()

*Create a random id to be used in conjunction with lbt\_get\_jms\_msg\_id for JMS compatibility.*

- LBMEExpDLL char \* **lbt\_get\_jms\_msg\_id** (lbt\_uint64\_t source\_id, lbt\_uint64\_t seqno\_id, char \*topic)

*Create JMS message ID.*

### 8.1.1 Detailed Description

#### Author:

Todd L. Montgomery - Informatica Corporation.

#### VersIdn:

//UMprod/REL\_5\_3\_6/29West/lbt/src/lib/lbt/lbt.h#2

The Ultra Messaging (UM) API Description. Included are types, constants, and functions related to the API. Contents are subject to change.

All of the documentation and software included in this and any other Informatica Ultra Messaging Releases Copyright (C) Informatica. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted only as covered by the terms of a valid software license agreement with Informatica.

Copyright (C) 2004-2014, Informatica Corporation. All Rights Reserved.

THE SOFTWARE IS PROVIDED "AS IS" AND INFORMATICA DISCLAIMS ALL WARRANTIES EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. INFORMATICA DOES NOT WARRANT THAT USE OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE. INFORMATICA SHALL NOT, UNDER ANY CIRCUMSTANCES, BE LIABLE TO LICENSEE FOR LOST PROFITS, CONSEQUENTIAL, INCIDENTAL, SPECIAL OR INDIRECT DAMAGES ARISING OUT OF OR RELATED TO THIS AGREEMENT OR THE TRANSACTIONS CONTEMPLATED HEREUNDER, EVEN IF INFORMATICA HAS BEEN APPRISED OF THE LIKELIHOOD OF SUCH DAMAGES.

### 8.1.2 Define Documentation

#### 8.1.2.1 **#define LBM\_ASYNC\_OP\_INFO\_FLAG\_FIRST 0x2**

`lbm_async_operation_info_t` flag. This is the very first notification for this particular asynchronous operation.

#### 8.1.2.2 **#define LBM\_ASYNC\_OP\_INFO\_FLAG\_INLINE 0x1**

`lbm_async_operation_info_t` flag. Asynchronous operation callback is being called directly inline from within an API call.

#### 8.1.2.3 **#define LBM\_ASYNC\_OP\_INFO\_FLAG\_LAST 0x4**

`lbm_async_operation_info_t` flag. This is the very last notification for this particular asynchronous operation.

#### 8.1.2.4 **#define LBM\_ASYNC\_OP\_INFO\_FLAG\_ONLY (LBM\_ASYNC\_OP\_INFO\_FLAG\_FIRST | LBM\_ASYNC\_OP\_INFO\_FLAG\_LAST)**

`lbm_async_operation_info_t` flag. This is the only notification that will be delivered for this particular asynchronous operation.

#### 8.1.2.5 **#define LBM\_ASYNC\_OP\_INVALID\_HANDLE 0**

Invalid asynchronous operation handle.

#### 8.1.2.6 **#define LBM\_ASYNC\_OP\_STATUS\_CANCELED 130**

Asynchronous operation status code. Overall operation has been successfully canceled. This is a terminal status code.

#### 8.1.2.7 **#define LBM\_ASYNC\_OP\_STATUS\_COMPLETE 128**

Asynchronous operation status code. Overall operation has completed successfully. This is a terminal status code.

#### 8.1.2.8 **#define LBM\_ASYNC\_OP\_STATUS\_ERROR 129**

Asynchronous operation status code. Overall operation has failed. This is a terminal status code.

**8.1.2.9 #define LBM\_ASYNC\_OP\_STATUS\_IN\_PROGRESS 1**

Asynchronous operation status code. Overall operation is still in progress.

**8.1.2.10 #define LBM\_ASYNC\_OP\_TYPE\_CTX\_UMQ\_QUEUE\_TOPIC\_LIST 1**

Asynchronous operation type. UMQ queue topic list.

**8.1.2.11 #define LBM\_ASYNC\_OP\_TYPE\_RCV\_UMQ\_QUEUE\_MSG\_LIST 2**

Asynchronous operation type. UMQ queue message list.

**8.1.2.12 #define LBM\_ASYNC\_OP\_TYPE\_RCV\_UMQ\_QUEUE\_MSG\_RETRIEVE 3**

Asynchronous operation type. UMQ queue message retrieve.

**8.1.2.13 #define LBM\_ASYNC\_OPERATION\_CANCEL\_FLAG\_NONBLOCK 0x1**

lbm\_async\_operation\_cancel flag. Do not block if the operation cannot be immediately canceled.

**8.1.2.14 #define LBM\_ASYNC\_OPERATION\_STATUS\_FLAG\_NONBLOCK 0x1**

lbm\_async\_operation\_status flag. Do not block if the operation's status cannot be retrieved immediately.

**8.1.2.15 #define LBM\_CHAIN\_ELEM\_APPHDR 0x4**

Element is a non-chain app header

**8.1.2.16 #define LBM\_CHAIN\_ELEM\_CHANNEL\_NUMBER 0x1**

Element is a channel number in network byte order

**8.1.2.17 #define LBM\_CHAIN\_ELEM\_GW\_INFO 0x3**

Element is gateway information

**8.1.2.18 #define LBM\_CHAIN\_ELEM\_HF\_SQN 0x2**

Element is a hot-failover sequence number in network byte order

**8.1.2.19 #define LBM\_CHAIN\_ELEM\_PROPERTIES\_LENGTH 0x6**

Element is the offset of a serialized properties object within an LBM message

**8.1.2.20 #define LBM\_CHAIN\_ELEM\_USER\_DATA 0x5**

Element is user data with no byte-order transformation applied

**8.1.2.21 #define LBM\_CONTEXT\_EVENT\_UMQ\_INSTANCE\_LIST\_NOTIFICATION 4**

Type of context event. For UMQ only, means queue instance list has changed. Event holds information string.

**8.1.2.22 #define LBM\_CONTEXT\_EVENT\_UMQ\_REGISTRATION\_COMPLETE\_EX 1**

Type of context event. For UMQ only, means registration of context complete. Event data holds Queue information, Registration ID, etc.

**8.1.2.23 #define LBM\_CONTEXT\_EVENT\_UMQ\_REGISTRATION\_COMPLETE\_EX\_FLAG\_QUORUM 0x1**

Registration completed with only quorum reached.

**8.1.2.24 #define LBM\_CONTEXT\_EVENT\_UMQ\_REGISTRATION\_ERROR 3**

Type of context event. For UMQ only, means registration of context failed with an error. Event data holds error string.

**8.1.2.25 #define LBM\_CONTEXT\_EVENT\_UMQ\_REGISTRATION\_SUCCESS\_EX 2**

Type of context event. For UMQ only, means registration of context successful with specific Queue instance. Event data holds Queue instance information, etc.

**8.1.2.26 #define LBM\_DAEMON\_EVENT\_CONNECT\_ERROR 2**

UM daemon event. Connected could not complete successfully (info valid)

**8.1.2.27 #define LBM\_DAEMON\_EVENT\_CONNECT\_TIMEOUT 4**

UM daemon event. Connection to daemon timed out

**8.1.2.28 #define LBM\_DAEMON\_EVENT\_CONNECTED 1**

UM daemon event. Connected successfully to daemon (info not valid)

**8.1.2.29 #define LBM\_DAEMON\_EVENT\_DISCONNECTED 3**

UM daemon event. Connection to daemon aborted (info not valid)

**8.1.2.30 #define LBM\_EDAEMONCONN 7**

[lbm\\_errnum\(\)](#) value. UM daemon connection not connected.

**8.1.2.31 #define LBM\_EINPROGRESS 10**

[lbm\\_errnum\(\)](#) value. Operation in progress.

**8.1.2.32 #define LBM\_EINVAL 1**

[lbm\\_errnum\(\)](#) value. An invalid argument was passed.

**8.1.2.33 #define LBM\_EMSG\_SELECTOR 14**

[lbm\\_errnum\(\)](#) Error parsing message selector.

**8.1.2.34 #define LBM\_ENO\_QUEUE\_REG 11**

[lbm\\_errnum\(\)](#) The queue is not fully registered.

**8.1.2.35 #define LBM\_ENO\_STORE\_REG 12**

[lbm\\_errnum\(\)](#) The store is not fully registered.

**8.1.2.36 #define LBM\_ENOMEM 3**

[lbm\\_errnum\(\)](#) value. Operation could not be completed due to memory allocation error.

**8.1.2.37 #define LBM\_EOP 4**

[lbm\\_errnum\(\)](#) value. Operation was invalid due to error in internal processing.

**8.1.2.38 #define LBM\_EOPNOTSUPP 9**

[lbm\\_errnum\(\)](#) value. Operation is not supported.

**8.1.2.39 #define LBM\_EOS 5**

[lbm\\_errnum\(\)](#) value. Operation failed due to unrecoverable OS system call error.

**8.1.2.40 #define LBM\_ETIMEDOUT 6**

[lbm\\_errnum\(\)](#) value. Operation timed out waiting to complete.

**8.1.2.41 #define LBM\_EUMENOREG 8**

[lbm\\_errnum\(\)](#) value. Registration not completed.

**8.1.2.42 #define LBM\_EVENT\_QUEUE\_BLOCK 0xFFFFFFFF**

Value passed to `lbm_event_dispatch` to ask it to block

**8.1.2.43 #define LBM\_EVENT\_QUEUE\_DELAY\_WARNING 0x2**

event queue monitor event type. Warning of excessive delay for event.

**8.1.2.44 #define LBM\_EVENT\_QUEUE\_ENQUEUE\_NOTIFICATION 0x3**

event queue monitor event type. Notification of something being added to queue.

**8.1.2.45 #define LBM\_EVENT\_QUEUE\_POLL 0x0**

Value passed to `lbt_event_dispatch` to ask it to poll

**8.1.2.46 #define LBM\_EVENT\_QUEUE\_SIZE\_WARNING 0x1**

event queue monitor event type. Warning of event queue size.

**8.1.2.47 #define LBM\_EWOULDBLOCK 2**

`lbt_errnum()` value. Function would block, but object is set to be nonblocking.

**8.1.2.48 #define LBM\_FD\_EVENT\_ACCEPT 0x8**

FD event. Accept connection (TCP) indication on file descriptor/socket

**8.1.2.49 #define LBM\_FD\_EVENT\_ALL 0x3f**

FD event. All events indication on file descriptor/socket

**8.1.2.50 #define LBM\_FD\_EVENT\_CLOSE 0x10**

FD event. Close indication on file descriptor/socket

**8.1.2.51 #define LBM\_FD\_EVENT\_CONNECT 0x20**

FD event. Connected indication on file descriptor/socket

**8.1.2.52 #define LBM\_FD\_EVENT\_EXCEPT 0x4**

FD event. Exception (OOB/URG) indication on file descriptor/socket

**8.1.2.53 #define LBM\_FD\_EVENT\_READ 0x1**

FD event. Read indication on file descriptor/socket

**8.1.2.54 #define LBM\_FD\_EVENT\_WRITE 0x2**

FD event. Write indication on file descriptor/socket

**8.1.2.55 #define LBM\_FLIGHT\_SIZE\_TYPE\_ULB 0x2**

Specify a ULB flight size

**8.1.2.56 #define LBM\_FLIGHT\_SIZE\_TYPE\_UME 0x1**

Specify a UME flight size

**8.1.2.57 #define LBM\_FLIGHT\_SIZE\_TYPE\_UMQ 0x3**

Specify a UMQ flight size

**8.1.2.58 #define LBM\_LOG\_ALERT 2**

log level. Alert

**8.1.2.59 #define LBM\_LOG\_CRIT 3**

log level. Critical

**8.1.2.60 #define LBM\_LOG\_DEBUG 8**

log level. Debugging information

**8.1.2.61 #define LBM\_LOG\_EMERG 1**

log level. Emergency

**8.1.2.62 #define LBM\_LOG\_ERR 4**

log level. Error

**8.1.2.63 #define LBM\_LOG\_INFO 7**

log level. Informational

**8.1.2.64 #define LBM\_LOG\_NOTICE 6**

log level. Notice

**8.1.2.65 #define LBM\_LOG\_WARNING 5**

log level. Warning

**8.1.2.66 #define LBM\_MSG\_BOS 20**

lbt\_msg\_t type. Beginning of Transport Session (source connection established) (data received).

**8.1.2.67 #define LBM\_MSG\_COMPLETE\_BATCH 0x3**

Flag passed to a source send call E.G. ([lbt\\_src\\_send\(\)](#), [lbt\\_src\\_send\\_ex\(\)](#) etc) : Message constitutes a complete batch and should be sent to the implicit batching buffer.

**8.1.2.68 #define LBM\_MSG\_DATA 0**

lbt\_msg\_t type. Data message, Message is composed of user data.

**8.1.2.69 #define LBM\_MSG\_END\_BATCH 0x2**

Flag passed to a source send call E.G. ([lbt\\_src\\_send\(\)](#), [lbt\\_src\\_send\\_ex\(\)](#) etc) : Message ends a batch of messages

**8.1.2.70 #define LBM\_MSG\_EOS 1**

lbt\_msg\_t type. End of Transport Session (connection closed to source) (no further data).

**8.1.2.71 #define LBM\_MSG\_FLAG\_DELIVERY\_LATENCY 0x200**

reserved for future use

**8.1.2.72 #define LBM\_MSG\_FLAG\_END\_BATCH 0x2**

lbt\_msg\_t flags. Message ends a batch.

**8.1.2.73 #define LBM\_MSG\_FLAG\_HF\_32 0x800**

lbm\_msg\_t flags. Message contains a 32 bit hot failover sequence number

**8.1.2.74 #define LBM\_MSG\_FLAG\_HF\_64 0x1000**

lbm\_msg\_t flags. Message contains a 64 bit hot failover sequence number.

**8.1.2.75 #define LBM\_MSG\_FLAG\_HF\_DUPLICATE 0x20**

lbm\_msg\_t flags. Message is a Hot Failover duplicate message.

**8.1.2.76 #define LBM\_MSG\_FLAG\_HF\_OPTIONAL 0x400**

lbm\_msg\_t flags. Message is a Hot Failover optional message.

**8.1.2.77 #define LBM\_MSG\_FLAG\_HF\_PASS\_THROUGH 0x4**

lbm\_msg\_t flags. Message is a passed-through Hot Failover message.

**8.1.2.78 #define LBM\_MSG\_FLAG\_IMMEDIATE 0x10**

lbm\_msg\_t flags. Message is an immediate message.

**8.1.2.79 #define LBM\_MSG\_FLAG\_NUMBERED\_CHANNEL 0x1**

lbm\_channel\_info\_t flags. Message was sent on a numbered channel

**8.1.2.80 #define LBM\_MSG\_FLAG\_OTR 0x2000**

lbm\_msg\_t flags. Message was recovered via OTR

**8.1.2.81 #define LBM\_MSG\_FLAG\_RETRANSMIT 0x8**

lbm\_msg\_t flags. Message is a retransmission.

**8.1.2.82 #define LBM\_MSG\_FLAG\_START\_BATCH 0x1**

lbm\_msg\_t flags. Message starts a batch.

**8.1.2.83 #define LBM\_MSG\_FLAG\_TOPICLESS 0x100**

lbt\_msg\_t flags. Message has no topic.

**8.1.2.84 #define LBM\_MSG\_FLAG\_UME\_RETRANSMIT 0x8**

lbt\_msg\_t flags. Message is a UMP retransmission.

**8.1.2.85 #define LBM\_MSG\_FLAG\_UMQ\_REASSIGNED 0x40**

lbt\_msg\_t flags. Message is a UMQ message that has been re-assigned at least once.

**8.1.2.86 #define LBM\_MSG\_FLAG\_UMQ\_RESUBMITTED 0x80**

lbt\_msg\_t flags. Message is a UMQ message that has been resubmitted at least once.

**8.1.2.87 #define LBM\_MSG\_FLUSH 0x4**

Flag passed to a source send call E.G. ([lbt\\_src\\_send\(\)](#), [lbt\\_src\\_send\\_ex\(\)](#) etc) : Message is to be sent ASAP (not implicitly or explicitly batched). This also flushes waiting messages that were explicitly or implicitly batched.

**8.1.2.88 #define LBM\_MSG\_HF\_RESET 27**

lbt\_msg\_t type. Hot-failover reset message was handled. UMS is now expecting msg->hf\_sequence\_number as the next non-reset hot-failover message.

**8.1.2.89 #define LBM\_MSG iov\_GATHER 0x40**

Flag passed to a source send vectored call E.G. ([lbt\\_src\\_sendv\(\)](#), [lbt\\_src\\_sendv\\_ex\(\)](#) etc) : iovec elements should be gather into a single message

**8.1.2.90 #define LBM\_MSG\_NO\_SOURCE\_NOTIFICATION 6**

lbt\_msg\_t type. Notification that no source has been found for topic. Still querying for topic source.

**8.1.2.91 #define LBM\_MSG\_PROPERTIES\_MAX\_NAMELEN 250**

Maximum size for the name of a message property

**8.1.2.92 #define LBM\_MSG\_PROPERTY\_BOOLEAN 0x1**

Message property of boolean type

**8.1.2.93 #define LBM\_MSG\_PROPERTY\_BYTE 0x2**

Message property of byte type

**8.1.2.94 #define LBM\_MSG\_PROPERTY\_DOUBLE 0x7**

Message property of double type

**8.1.2.95 #define LBM\_MSG\_PROPERTY\_FLOAT 0x6**

Message property of float type

**8.1.2.96 #define LBM\_MSG\_PROPERTY\_INT 0x4**

Message property of int type (4 bytes)

**8.1.2.97 #define LBM\_MSG\_PROPERTY\_LONG 0x5**

Message property of long type (8 bytes)

**8.1.2.98 #define LBM\_MSG\_PROPERTY\_NONE 0x0**

Message property with no type (used to indicate an iterator has reached the last element)

**8.1.2.99 #define LBM\_MSG\_PROPERTY\_SHORT 0x3**

Message property of short type (2 bytes)

**8.1.2.100 #define LBM\_MSG\_PROPERTY\_STRING 0x8**

Message property of string type

**8.1.2.101 #define LBM\_MSG\_REQUEST 2**

lbm\_msg\_t type. Request message from source.

**8.1.2.102 #define LBM\_MSG\_RESPONSE 3**

lbm\_msg\_t type. Response message from requestee

**8.1.2.103 #define LBM\_MSG\_START\_BATCH 0x1**

Flag passed to a source send call E.G. ([lbm\\_src\\_send\(\)](#), [lbm\\_src\\_send\\_ex\(\)](#) etc) : Message starts a batch

**8.1.2.104 #define LBM\_MSG\_UME\_DEREGISTRATION\_COMPLETE\_-  
EX 13**

lbm\_msg\_t type. UMP receiver notification of deregistration complete.

**8.1.2.105 #define LBM\_MSG\_UME\_DEREGISTRATION\_SUCCESS\_EX 12**

lbm\_msg\_t type. UMP receiver notification of deregistration success. Data holds registration IDs, etc.

**8.1.2.106 #define LBM\_MSG\_UME\_DEREGISTRATION\_SUCCESS\_EX\_-  
FLAG\_RPP 0x1**

Deregistration was flagged as coming from a RPP store

**8.1.2.107 #define LBM\_MSG\_UME\_REGISTRATION\_CHANGE 9**

lbm\_msg\_t type. UMP receiver notification of source registration change. Data holds info message.

**8.1.2.108 #define LBM\_MSG\_UME\_REGISTRATION\_COMPLETE\_EX 11**

lbm\_msg\_t type. UMP receiver notification of registration completion. Data holds sequence number and flags, etc.

**8.1.2.109 #define LBM\_MSG\_UME\_REGISTRATION\_COMPLETE\_EX\_-  
FLAG\_QUORUM 0x1**

Registration completed with only quorum reached.

**8.1.2.110 #define LBM\_MSG\_UME\_REGISTRATION\_COMPLETE\_EX\_-  
FLAG\_RXREQMAX 0x2**

Registration completed with RX REQ maximum used.

**8.1.2.111 #define LBM\_MSG\_UME\_REGISTRATION\_ERROR 7**

lbm\_msg\_t type. UMP receiver registration encountered an error. Data holds error message.

**8.1.2.112 #define LBM\_MSG\_UME\_REGISTRATION\_SUCCESS 8**

lbm\_msg\_t type. UMP receiver registration successful. Data holds registration IDs.

**8.1.2.113 #define LBM\_MSG\_UME\_REGISTRATION\_SUCCESS\_EX 10**

lbm\_msg\_t type. UMP receiver registration successful for a store (extended form). Data holds registration IDs, etc.

**8.1.2.114 #define LBM\_MSG\_UME\_REGISTRATION\_SUCCESS\_EX\_-  
FLAG\_NOCACHE 0x2**

Registration was flagged as coming from a store that is configured to not cache data.

**8.1.2.115 #define LBM\_MSG\_UME\_REGISTRATION\_SUCCESS\_EX\_-  
FLAG\_OLD 0x1**

Registration was flagged as an old receiver by the store. An old receiver is one the store had cached.

**8.1.2.116 #define LBM\_MSG\_UME\_REGISTRATION\_SUCCESS\_EX\_-  
FLAG\_RPP 0x4**

Registration was flagged as coming from a store that has allowed a RPP receiver

**8.1.2.117 #define LBM\_MSG\_UMQ\_DEREGISTRATION\_COMPLETE\_-  
EX 19**

lbm\_msg\_t type. UMQ receiver notification of de-registration completion. Data holds Queue information, etc.

**8.1.2.118 #define LBM\_MSG\_UMQ\_DEREGISTRATION\_COMPLETE\_-  
EX\_FLAG\_ULB 0x1**

Deregistration completed for UMQ ULB source.

**8.1.2.119 #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNED\_EX 24**

lbm\_msg\_t type. UMQ receiver notification of beginning of index.

**8.1.2.120 #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNED\_EX\_FLAG\_-  
REQUESTED 0x2**

Beginning of index assignment that was requested by receiver.

**8.1.2.121 #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNED\_EX\_FLAG\_-  
ULB 0x1**

Beginning of index from ULB source.

**8.1.2.122 #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNMENT\_-  
ELIGIBILITY\_ERROR 21**

lbm\_msg\_t type. UMQ receiver index assignment start/stop encountered an error. Data holds error message.

**8.1.2.123 #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNMENT\_-  
ELIGIBILITY\_START\_COMPLETE\_EX 22**

lbm\_msg\_t type. UMQ receiver notification of beginning of index assignment eligibility or index assignment. Data holds index information, etc.

**8.1.2.124 #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNMENT\_-  
ELIGIBILITY\_START\_COMPLETE\_EX\_FLAG\_ULB 0x1**

Index assignment started for ULB source.

**8.1.2.125 #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNMENT\_-  
ELIGIBILITY\_STOP\_COMPLETE\_EX 23**

lbm\_msg\_t type. UMQ receiver notification of end of index assignment eligibility or index assignment. Data holds index information, etc.

**8.1.2.126 #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNMENT\_-  
ELIGIBILITY\_STOP\_COMPLETE\_EX\_FLAG\_ULB 0x1**

Index assignment stopped for ULB source.

**8.1.2.127 #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNMENT\_ERROR 26**

lbm\_msg\_t type. UMQ receiver notification of an index assignment error.

**8.1.2.128 #define LBM\_MSG\_UMQ\_INDEX\_RELEASED\_EX 25**

lbm\_msg\_t type. UMQ receiver notification of end of index.

**8.1.2.129 #define LBM\_MSG\_UMQ\_INDEX\_RELEASED\_EX\_FLAG\_-  
ULB 0x1**

End of index from ULB source.

**8.1.2.130 #define LBM\_MSG\_UMQ\_REASSIGN\_FLAG\_DISCARD 0x1**

Instead of requesting reassignment, request the message be discarded.

**8.1.2.131 #define LBM\_MSG\_UMQ\_REGISTRATION\_COMPLETE\_EX 18**

lbm\_msg\_t type. UMQ receiver notification of registration completion. Data holds Queue information, assignment ID, etc.

**8.1.2.132 #define LBM\_MSG\_UMQ\_REGISTRATION\_COMPLETE\_EX\_-  
FLAG\_QUORUM 0x1**

Registration completed with only quorum reached.

**8.1.2.133 #define LBM\_MSG\_UMQ\_REGISTRATION\_COMPLETE\_EX\_-  
FLAG\_ULB 0x2**

Registration completed for UMQ ULB source.

**8.1.2.134 #define LBM\_MSG\_UMQ\_REGISTRATION\_ERROR 16**

lbt\_msg\_t type. UMQ receiver registration encountered an error. Data holds error message.

**8.1.2.135 #define LBM\_MSG\_UNRECOVERABLE\_LOSS 4**

lbt\_msg\_t type. Missing message detected and not recovered in given time.

**8.1.2.136 #define LBM\_MSG\_UNRECOVERABLE\_LOSS\_BURST 5**

lbt\_msg\_t type. Missing burst of messages detected and not recovered.

**8.1.2.137 #define LBM\_RCV\_BLOCK 0x20**

reserved for future use

**8.1.2.138 #define LBM\_RCV\_BLOCK\_TEMP 0x10**

reserved for future use

**8.1.2.139 #define LBM\_RCV\_NONBLOCK 0x8**

reserved for future use

**8.1.2.140 #define LBM\_SRC\_BLOCK 0x20**

Flag passed to a source send call E.G. ([lbt\\_src\\_send\(\)](#), [lbt\\_src\\_send\\_ex\(\)](#) etc) : Block the caller indefinitely until the message is sent. (This behavior is the default if neither LBM\_SRC\_NONBLOCK nor LBM\_SRC\_BLOCK are supplied.)

**8.1.2.141 #define LBM\_SRC\_BLOCK\_TEMP 0x10**

reserved for future use

**8.1.2.142 #define LBM\_SRC\_COST\_FUNCTION\_REJECT 0xffffffff**

Source cost function return value to indicate this source should be permanently rejected.

**8.1.2.143 #define LBM\_SRC\_EVENT\_CONNECT 1**

Type of source event. This event indicates that the first or initial receiver of a receiving application has joined a unicast transport session. UM delivers this event if it has mapped a source object to a unicast transport session (TCP, LBT-RU, LBT-IPC, or LBT-RDMA) and then the first receiver joins the transport session. If several sources map to the transport session, UM delivers this event multiple times, once for each source. The initial receiver can be subscribed to any of the sources topics. Subsequent receivers that join the transport session do not trigger additional events. However, additional receiving applications (contexts) may also join the transport session and UM delivers the event for the first or initial receiver of each additional application.

**8.1.2.144 #define LBM\_SRC\_EVENT\_DAEMON\_CONFIRM 4**

Type of source event. For UM daemon usage only, means daemon has confirmed src created

**8.1.2.145 #define LBM\_SRC\_EVENT\_DISCONNECT 2**

Type of source event. This event indicates that the last or final receiver of a receiving application has left a unicast transport session. UM delivers this event if it has mapped a source object to a unicast transport session (TCP, LBT-RU, LBT-IPC, or LBT-RDMA) and then the last receiver leaves the transport session. If several sources map to the transport session, UM delivers this event multiple times, once for each source. The final receiver can be subscribed to any of the sources topics. Additional receiving applications (contexts) may also leave the transport session and UM delivers the event for the last or final receiver of each additional application.

**8.1.2.146 #define LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION 29**

Type of source event. For UMP, UMQ, and/or ULB, informs the application of a change in state for a specified flight size. Event data holds state information.

**8.1.2.147 #define LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION\_-  
STATE\_OVER 0x1**

Messages in flight has exceeded the threshold

**8.1.2.148 #define LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION\_-  
STATE\_UNDER 0x2**

Messages in flight is now below the threshold

**8.1.2.149 #define LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION\_-  
TYPE\_ULB 0x2**

Specifies a ULB flight size

**8.1.2.150 #define LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION\_-  
TYPE\_UME 0x1**

Specifies a UMP flight size

**8.1.2.151 #define LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION\_-  
TYPE\_UMQ 0x3**

Specifies a UMQ flight size

**8.1.2.152 #define LBM\_SRC\_EVENT\_SEQUENCE\_NUMBER\_INFO 15**

Type of source event. Informs the application the sequence numbers used with a message. Event data holds sequence number data. This event is generated only when using the "lbm\_src\_send\_ex()" API's with the LBM\_SRC\_SEND\_-EX\_FLAG\_SEQUENCE\_NUMBER\_INFO flag or LBM\_SRC\_SEND\_EX\_FLAG\_-SEQUENCE\_NUMBER\_INFO\_FRAGONLY flag.

**8.1.2.153 #define LBM\_SRC\_EVENT\_UME\_DELIVERY\_-  
CONFIRMATION 8**

Type of source event. For UMP only, means UMP Confirmed Delivery of Message from receiver. Event data holds ACK information.

**8.1.2.154 #define LBM\_SRC\_EVENT\_UME\_DELIVERY\_-  
CONFIRMATION\_EX 14**

Type of source event. For UMP only, means UMP Confirmed Delivery of Message from receiver (extended form). Event data holds ACK information.

**8.1.2.155 #define LBM\_SRC\_EVENT\_UME\_DELIVERY\_-  
CONFIRMATION\_EX\_FLAG\_EXACK 0x8**

Confirmation received with Explicit ACK (EXACK) flagged.

**8.1.2.156 #define LBM\_SRC\_EVENT\_UME\_DELIVERY\_-  
CONFIRMATION\_EX\_FLAG\_OOD 0x4**

Confirmation received with Out-of-Order Delivery (OOD) flagged.

**8.1.2.157 #define LBM\_SRC\_EVENT\_UME\_DELIVERY\_-  
CONFIRMATION\_EX\_FLAG\_UNIQUEACKS 0x1**

Confirmation received for specified number of unique ACKs.

**8.1.2.158 #define LBM\_SRC\_EVENT\_UME\_DELIVERY\_-  
CONFIRMATION\_EX\_FLAG\_UREGID 0x2**

Confirmation received with User Specified Rcv Registration ID flagged.

**8.1.2.159 #define LBM\_SRC\_EVENT\_UME\_DELIVERY\_-  
CONFIRMATION\_EX\_FLAG\_WHOLE\_MESSAGE\_-  
CONFIRMED 0x10**

Whole message (each fragment) has been confirmed

**8.1.2.160 #define LBM\_SRC\_EVENT\_UME\_DEREGISTRATION\_-  
COMPLETE\_EX 32**

Type of source event. For UMP only, means deregistration of source complete (extended form).

**8.1.2.161 #define LBM\_SRC\_EVENT\_UME\_DEREGISTRATION\_-  
SUCCESS\_EX 31**

Type of source event. For UMP only, means deregistration of source successful (extended form).

**8.1.2.162 #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_RECLAIMED 10**

Type of source event. For UMP only, means message is being reclaimed. Event data holds ACK information.

**8.1.2.163 #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_RECLAIMED\_-  
EX 30**

Type of source event. Message is being reclaimed. Event data holds ACK information.

**8.1.2.164 #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_RECLAIMED\_EX\_-  
FLAG\_FORCED 0x1**

Reclaim notification is the result of a forced reclaim

**8.1.2.165 #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE 7**

Type of source event. For UMP only, means UMP ACK from store indicates message is stable. Event data holds ACK information.

**8.1.2.166 #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE\_EX 13**

Type of source event. For UMP only, means UMP ACK from store indicates message is stable (extended form). Event data holds ACK information.

**8.1.2.167 #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE\_EX\_-  
FLAG\_INTERGROUP\_STABLE 0x2**

Message stable for intergroup stability

**8.1.2.168 #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE\_EX\_-  
FLAG\_INTRAGROUP\_STABLE 0x1**

Message stable for intragroup stability

**8.1.2.169 #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE\_EX\_-  
FLAG\_STABLE 0x4**

Message is stable according to behavior desired

**8.1.2.170 #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE\_EX\_-  
FLAG\_STORE 0x8**

Message stable information has active store information

**8.1.2.171 #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE\_EX\_-  
FLAG\_USER 0x20**

Message stabilized via lbm\_ume\_src\_msg\_stable API

**8.1.2.172 #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE\_EX\_-  
FLAG\_WHOLE\_MESSAGE\_STABLE 0x10**

Whole message (each fragment) is stable

**8.1.2.173 #define LBM\_SRC\_EVENT\_UME\_REGISTRATION\_-  
COMPLETE\_EX 12**

Type of source event. For UMP only, means registration of source complete (extended form). Event data holds sequence number, flags, etc.

**8.1.2.174 #define LBM\_SRC\_EVENT\_UME\_REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_QUORUM 0x1**

Registration completed with only quorum reached.

**8.1.2.175 #define LBM\_SRC\_EVENT\_UME\_REGISTRATION\_ERROR 5**

Type of source event. For UMP only, means registration of source failed with an error. Event data holds error string.

**8.1.2.176 #define LBM\_SRC\_EVENT\_UME\_REGISTRATION\_SUCCESS 6**

Type of source event. For UMP only, means registration of source successful. Event data holds registration info

**8.1.2.177 #define LBM\_SRC\_EVENT\_UME\_REGISTRATION\_SUCCESS\_-  
EX 11**

Type of source event. For UMP only, means registration of source successful (extended form). Event data holds registration info

**8.1.2.178 #define LBM\_SRC\_EVENT\_UME\_REGISTRATION\_SUCCESS\_-  
EX\_FLAG\_NOACKS 0x2**

Registration was flagged as coming from a store that is configured to not send ACKs for stability (no-cache store)

**8.1.2.179 #define LBM\_SRC\_EVENT\_UME\_REGISTRATION\_SUCCESS\_-  
EX\_FLAG\_OLD 0x1**

Registration was flagged as an old source by the store. An old source is one the store had cached.

**8.1.2.180 #define LBM\_SRC\_EVENT\_UME\_REGISTRATION\_SUCCESS\_-  
EX\_FLAG\_RPP 0x4**

Registration was flagged as coming from a store that allows and has accepted RPP persistent topics

**8.1.2.181 #define LBM\_SRC\_EVENT\_UME\_STORE\_UNRESPONSIVE 9**

Type of source event. For UMP only, means store has not been active within timeout. Event data holds info string.

**8.1.2.182 #define LBM\_SRC\_EVENT\_UMQ\_MESSAGE\_ID\_INFO 17**

Type of source event. For UMQ only, informs the application of the Message ID assigned with a message. Event data holds Message ID, etc.

**8.1.2.183 #define LBM\_SRC\_EVENT\_UMQ\_MESSAGE\_STABLE\_EX 19**

Type of source event. For UMQ only, means UMQ ACK from queue indicates message is stable. Event data holds ACK information.

**8.1.2.184 #define LBM\_SRC\_EVENT\_UMQ\_MESSAGE\_STABLE\_EX\_-  
FLAG\_INTEGROUP\_STABLE 0x2**

Message stable for intergroup stability

**8.1.2.185 #define LBM\_SRC\_EVENT\_UMQ\_MESSAGE\_STABLE\_EX\_-  
FLAG\_INTRAGROUP\_STABLE 0x1**

Message stable for intragroup stability

**8.1.2.186 #define LBM\_SRC\_EVENT\_UMQ\_MESSAGE\_STABLE\_EX\_-  
FLAG\_STABLE 0x4**

Message is stable according to behavior desired

**8.1.2.187 #define LBM\_SRC\_EVENT\_UMQ\_MESSAGE\_STABLE\_EX\_-  
FLAG\_USER 0x8**

Message stabilized via the `lbm_umq_ctx_msg_stable` API

**8.1.2.188 #define LBM\_SRC\_EVENT\_UMQ\_REGISTRATION\_-  
COMPLETE\_EX 18**

Type of source event. For UMQ only, means registration of source complete. Event data holds Queue information, etc.

**8.1.2.189 #define LBM\_SRC\_EVENT\_UMQ\_REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_QUORUM 0x1**

Registration completed with only quorum reached.

**8.1.2.190 #define LBM\_SRC\_EVENT\_UMQ\_REGISTRATION\_ERROR 16**

Type of source event. For UMQ only, means registration of source failed with an error. Event data holds error string.

**8.1.2.191 #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_ASSIGNED\_-  
EX 20**

Type of source event. For UMQ ULB only, means message was assigned to a receiver. Event data holds message information.

**8.1.2.192 #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_-  
COMPLETE\_EX 23**

Type of source event. For UMQ ULB only, means message was completed processed on all application sets. Event data holds message information.

**8.1.2.193 #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_-  
CONSUMED\_EX 24**

Type of source event. For UMQ ULB only, means message was consumed by a receiver. Event data holds message information.

**8.1.2.194 #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_-  
REASSIGNED\_EX 21**

Type of source event. For UMQ ULB only, means message was reassigned from a receiver. Event data holds message information.

**8.1.2.195 #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_-  
REASSIGNED\_EX\_FLAG\_EXPLICIT 0x1**

reassignment is the result of the `lbt_msg_umq_reassign` API being called by a receiver

**8.1.2.196 #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_TIMEOUT\_-  
EX 22**

Type of source event. For UMQ ULB only, means message timed out and was end-of-lifed. Event data holds message information.

**8.1.2.197 #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_TIMEOUT\_-  
EX\_FLAG\_DISCARD 0x4**

timeout is the result of the `lbt_msg_umq_reassign` API being called with the `LBM_MSG_UMQ_REASSIGN_FLAG_DISCARD` flag set

**8.1.2.198 #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_TIMEOUT\_-  
EX\_FLAG\_EXPLICIT 0x2**

timeout is the result of the `lbt_msg_umq_reassign` API being called by a receiver

**8.1.2.199 #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_TIMEOUT\_-  
EX\_FLAG\_MAX\_REASSIGNS 0x8**

timeout is the result of hitting umq\_ulb\_application\_set\_message\_max\_reassignments number of assignments

**8.1.2.200 #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_TIMEOUT\_-  
EX\_FLAG\_TOTAL\_LIFETIME\_EXPIRED 0x1**

timeout is the result of the total lifetime expiring

**8.1.2.201 #define LBM\_SRC\_EVENT\_UMQ\_ULB\_RECEIVER\_-  
Deregistration\_Ex 26**

Type of source event. For UMQ ULB only, means receiver deregistered. Event data holds receiver information.

**8.1.2.202 #define LBM\_SRC\_EVENT\_UMQ\_ULB\_RECEIVER\_READY\_-  
Ex 27**

Type of source event. For UMQ ULB only, means receiver signalled ready for messages. Event data holds receiver information.

**8.1.2.203 #define LBM\_SRC\_EVENT\_UMQ\_ULB\_RECEIVER\_-  
Registration\_Ex 25**

Type of source event. For UMQ ULB only, means receiver registered. Event data holds receiver information.

**8.1.2.204 #define LBM\_SRC\_EVENT\_UMQ\_ULB\_RECEIVER\_TIMEOUT\_-  
Ex 28**

Type of source event. For UMQ ULB only, means receiver timed out and was end-of-lifed. Event data holds receiver information.

**8.1.2.205 #define LBM\_SRC\_EVENT\_WAKEUP 3**

Type of source event. Following earlier return of LBM\_EWOULDBLOCK, means source is ready for more sends

**8.1.2.206 #define LBM\_SRC\_EVENT\_WAKEUP\_FLAG\_MIM 0x2**

Unblocked source is a context-level multicast immediate mode source.

**8.1.2.207 #define LBM\_SRC\_EVENT\_WAKEUP\_FLAG\_NORMAL 0x1**

Unblocked source is a normal (or hot failover) source.

**8.1.2.208 #define LBM\_SRC\_EVENT\_WAKEUP\_FLAG\_REQUEST 0x8**

Unblocked source is a context-level request source.

**8.1.2.209 #define LBM\_SRC\_EVENT\_WAKEUP\_FLAG\_RESPONSE 0x10**

Unblocked source is a context-level response source.

**8.1.2.210 #define LBM\_SRC\_EVENT\_WAKEUP\_FLAG\_UIM 0x4**

Unblocked source is a context-level unicast immediate mode source.

**8.1.2.211 #define LBM\_SRC\_NONBLOCK 0x8**

Flag passed to a source send call E.G. ([lbt\\_src\\_send\(\)](#), [lbt\\_src\\_send\\_ex\(\)](#) etc)  
: If message could not be sent immediately return and error and signal LBM\_EWOULDBLOCK.

**8.1.2.212 #define LBM\_SRC\_SEND\_EX\_FLAG\_APPHDR\_CHAIN 0x8**

Send messages using an appheader chain

**8.1.2.213 #define LBM\_SRC\_SEND\_EX\_FLAG\_CHANNEL 0x20**

Send messages using supplied channel information

**8.1.2.214 #define LBM\_SRC\_SEND\_EX\_FLAG\_HF\_32 0x400**

Send message with the supplied 32 bit hot failover sequence number

**8.1.2.215 #define LBM\_SRC\_SEND\_EX\_FLAG\_HF\_64 0x800**

Send message with the supplied 64 bit hot failover sequence number

**8.1.2.216 #define LBM\_SRC\_SEND\_EX\_FLAG\_HF\_OPTIONAL 0x100**

Send messages marked as an optional message for Hot Failover

**8.1.2.217 #define LBM\_SRC\_SEND\_EX\_FLAG\_PROPERTIES 0x200**

Send message with the supplied messages properties

**8.1.2.218 #define LBM\_SRC\_SEND\_EX\_FLAG\_SEQUENCE\_NUMBER\_-  
INFO 0x2**

Inform application of the sequence numbers used for message

**8.1.2.219 #define LBM\_SRC\_SEND\_EX\_FLAG\_SEQUENCE\_NUMBER\_-  
INFO\_FRAGONLY 0x4**

Inform application of the sequence numbers used for message (fragmented messages only)

**8.1.2.220 #define LBM\_SRC\_SEND\_EX\_FLAG\_UME\_CLIENTD 0x1**

UMP client data pointer is valid

**8.1.2.221 #define LBM\_SRC\_SEND\_EX\_FLAG\_UMQ\_CLIENTD 0x1**

UMQ client data pointer is valid

**8.1.2.222 #define LBM\_SRC\_SEND\_EX\_FLAG\_UMQ\_INDEX 0x40**

Send messages associating them with the supplied UMQ Index

**8.1.2.223 #define LBM\_SRC\_SEND\_EX\_FLAG\_UMQ\_MESSAGE\_ID\_-  
INFO 0x10**

Inform application of the UMQ Message ID used for the message

**8.1.2.224 #define LBM\_SRC\_SEND\_EX\_FLAG\_UMQ\_TOTAL\_-LIFETIME 0x80**

umq\_msg\_total\_lifetime is valid

**8.1.2.225 #define LBM\_TOPIC\_RES\_REQUEST\_ADVERTISEMENT 0x04**

Flag passed to [lbt\\_context\\_topic\\_resolution\\_request\(\)](#) to request sources to re-advertise

**8.1.2.226 #define LBM\_TOPIC\_RES\_REQUEST\_QUERY 0x02**

Flag passed to [lbt\\_context\\_topic\\_resolution\\_request\(\)](#) to request receivers to query for source

**8.1.2.227 #define LBM\_TOPIC\_RES\_REQUEST\_RESERVED1 0x08**

reserved for internal use

**8.1.2.228 #define LBM\_TOPIC\_RES\_REQUEST\_WILDCARD\_QUERY 0x01**

Flag passed to [lbt\\_context\\_topic\\_resolution\\_request\(\)](#) to request wildcard receivers to query for source

**8.1.2.229 #define LBM\_TRANSPORT\_STAT\_DAEMON 0xFF**

Transport statistic type. UM Daemon is being used

**8.1.2.230 #define LBM\_TRANSPORT\_STAT\_LBTIPC LBM\_TRANSPORT\_TYPE\_LBTIPC**

Transport statistic type. LBT-IPC transport

**8.1.2.231 #define LBM\_TRANSPORT\_STAT\_LBTRDMA LBM\_TRANSPORT\_TYPE\_LBTRDMA**

Transport statistic type. LBT-RDMA transport

**8.1.2.232 #define LBM\_TRANSPORT\_STAT\_LBTRM LBM\_TRANSPORT\_-  
TYPE\_LBTRM**

Transport statistic type. LBT-RM transport

**8.1.2.233 #define LBM\_TRANSPORT\_STAT\_LBTRU LBM\_TRANSPORT\_-  
TYPE\_LBTRU**

Transport statistic type. LBT-RU transport

**8.1.2.234 #define LBM\_TRANSPORT\_STAT\_TCP LBM\_TRANSPORT\_-  
TYPE\_TCP**

Transport statistic type. TCP transport

**8.1.2.235 #define LBM\_TRANSPORT\_TYPE\_LBTIPC 0x40**

Transport type LBT-IPC.

**8.1.2.236 #define LBM\_TRANSPORT\_TYPE\_LBTRDMA 0x20**

Transport type LBT-RDMA.

**8.1.2.237 #define LBM\_TRANSPORT\_TYPE\_LBTRM 0x10**

Transport type LBT-RM.

**8.1.2.238 #define LBM\_TRANSPORT\_TYPE\_LBTRU 0x01**

Transport type LBT-RU.

**8.1.2.239 #define LBM\_TRANSPORT\_TYPE\_TCP 0x00**

Transport type TCP.

**8.1.2.240 #define LBM\_UMM\_INFO\_FLAGS\_USE\_SSL 0x1**

Use SSL for UMM daemon connections.

**8.1.2.241 #define LBM\_UMQ\_INDEX\_FLAG\_NUMERIC 0x1**

lbm\_umq\_index\_info\_t flags. Index is a 64-bit unsigned integer.

**8.1.2.242 #define LBM\_WILDCARD\_RCV\_PATTERN\_TYPE\_APP\_CB 3**

Application defined callback pattern type

**8.1.2.243 #define LBM\_WILDCARD\_RCV\_PATTERN\_TYPE\_PCRE 1**

PCRE (Perl Compatible Regular Expressions) pattern type

**8.1.2.244 #define LBM\_WILDCARD\_RCV\_PATTERN\_TYPE\_REGEX 2**

POSIX regex pattern type

### 8.1.3 Typedef Documentation

**8.1.3.1 typedef int(\*) lbm\_async\_operation\_function\_cb(lbm\_async\_operation\_info\_t \*opinfo, void \*clientd)****Parameters:**

*opinfo* Operation-specific results.

*clientd* Client data pointer supplied in the [lbm\\_async\\_operation\\_func\\_t](#) struct passed in to an asynchronous API call.

**Returns:**

0 for Success, -1 for Failure.

**8.1.3.2 typedef int(\*) lbm\_context\_event\_cb\_proc(lbm\_context\_t \*ctx, int event, void \*ed, void \*clientd)**

Set by the "context\_event\_function" context attribute. If this application callback is set without an event queue, it is called from the context thread and is limited in the API calls that it can make.

**Parameters:**

*ctx* Context object generating the event.

*event* Type of event.

*ed* Pointer to event data, content dependent on event type.

*clientd* Client data pointer supplied when setting "context\_event\_function" context attribute.

**Returns:**

0 always

**8.1.3.3 `typedef struct lbm_context_event_func_t_stct`  
`lbm_context_event_func_t`**

A struct used to set a context-level event callback and callback info.

**8.1.3.4 `typedef struct lbm_context_event_umq_registration_complete_ex_t_stct`  
`lbm_context_event_umq_registration_complete_ex_t`**

A structure used with UMQ receivers and sources to indicate successful context registration to quorum or to all queue instances involved.

**8.1.3.5 `typedef struct lbm_context_event_umq_registration_ex_t_stct`  
`lbm_context_event_umq_registration_ex_t`**

A structure used with UMQ receivers and sources to indicate successful context registration with an instance of the queue.

**8.1.3.6 `typedef struct lbm_context_rcv_immediate_msgs_func_t_stct`  
`lbm_context_rcv_immediate_msgs_func_t`**

A struct used to set the context-level topic-less immediate mode message receiver callback. If an event queue is specified, messages will be placed on the event queue; if evq is NULL, messages will be delivered directly from the context thread.

**8.1.3.7 `typedef int(*) lbm_context_src_cb_proc(lbm_context_t *ctx, int event,`  
`void *ed, void *clientd)`**

Set by the "source\_event\_function" context attribute. If this application callback is set without an event queue, it is called from the context thread and is limited in the API calls that it can make.

**Parameters:**

*ctx* Context object generating the event.

***ed*** Pointer to event data, content dependent on event type.

- For *event* == LBM\_SRC\_EVENT\_WAKEUP, *ed* should be re-cast as a (lbt\_src\_event\_wakeup\_t) and indicates which context-level source (or sources) has become un-blocked.
- For *event* == LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION, *ed* should be re-cast as a (lbt\_src\_event\_flight\_size\_notification\_t \*) to extract the flight size information.

***clientd*** Client data pointer supplied when setting "source\_event\_function" context attribute.

**Returns:**

0 always

#### **8.1.3.8 `typedef struct lbt_context_src_event_func_t_stct lbt_context_src_event_func_t`**

A struct used to set a context-level source event callback and callback info.

#### **8.1.3.9 `typedef struct lbt_context_stats_t_stct lbt_context_stats_t`**

This structure holds general context statistics for things like topic resolution and interaction with transports and applications.

#### **8.1.3.10 `typedef int(*) lbt_daemon_event_cb_proc(lbt_context_t *ctx, int event, const char *info, void *clientd)`**

Set by [lbt\\_context\\_create\(\)](#). Only used when operation\_mode is set to daemon. NOTE: this application callback is always made from the context thread, and is therefore limited in the UM API calls it can make. NOTE: daemon mode is no longer available; this definition is retained for backward compatibility only.

**Parameters:**

***ctx*** Context generating the event.

***event*** One of LBM\_DAEMON\_EVENT\_\* indicating event type.

***info*** Pointer to LBM-supplied string giving more information.

***clientd*** Client data pointer supplied in [lbt\\_context\\_create\(\)](#).

**Returns:**

0 always.

### 8.1.3.11 **typedef void(\*) lbm\_event\_queue\_cancel\_cb\_proc(int dispatch\_thrd, void \*clientd)**

Set by `lbm_*_delete_ex()`. NOTE: this application callback can be made from the context thread, and is therefore limited in the UM API calls it can make. The application is called after all events associated with the delete are canceled or completed.

**See also:**

[lbm\\_event\\_queue\\_cancel\\_cb\\_info\\_t](#)

**Parameters:**

*dispatch\_thrd* Indicates from where the callback is being called. This can be useful to the application to avoid deadlock.

- 1 - Called by dispatch thread (after the `lbm_*_delete_ex()` returned).
- 0 - Called directly by the `lbm_*_delete_ex()` function.

*clientd* Client data pointer supplied in the `lbm_event_queue_cancel_cb_info_t` passed to the `lbm_*_delete_ex()`.

**Returns:**

0 always.

### 8.1.3.12 **typedef int(\*) lbm\_event\_queue\_monitor\_proc(lbm\_event\_queue\_t \*evq, int event, size\_t evq\_size, lbm\_ulong\_t event\_delay\_usec, void \*clientd)**

Set by `lbm_event_queue_create()`. NOTE: this application callback can be made from the context thread, and is therefore limited in the UM API calls it can make. (Specifically, the context calls it for the enqueue notification.) Note that the one or more event queue options must be set to enable the use of event queue monitoring.

**Parameters:**

*evq* Event queue generating the event.

*event* One of `LBM_EVENT_QUEUE_*_WARNING` or `LBM_EVENT_QUEUE_ENQUEUE_NOTIFICATION`, depending on enabled options.

*evq\_size* Number of events currently in the queue.

*event\_delay\_usec* Number of microseconds the oldest event has been in the event queue. (Note, this will be the next event dispatched.)

*clientd* Client data pointer supplied in `lbm_event_queue_create()`.

**Returns:**

0 for success, -1 for failure.

**Note:**

The *event* parameter operates as both a value and a bitmask, and the monitor function may be called to indicate both a size and delay warning. The value of **LBM\_EVENT\_QUEUE\_ENQUEUE\_NOTIFICATION** is the same as the value of (**LBM\_EVENT\_QUEUE\_DELAY\_WARNING** | **LBM\_EVENT\_QUEUE\_SIZE\_WARNING**).

If *event* is **LBM\_EVENT\_QUEUE\_ENQUEUE\_NOTIFICATION**, *evq\_size* is 1, and *evq\_delay\_usec* is 0, the callback is due to an event being enqueued. To distinguish between an enqueue notification and a size or delay warning, the following code template may be used.

```
int event_queue_monitor_proc(lbm_event_queue_t * evq, int event,
    size_t evq_size, lbm_ulong_t event_delay_usec, void * clientd)
{
    if ((event == LBM_EVENT_QUEUE_ENQUEUE_NOTIFICATION) && (evq_size == 1) && (event_delay_usec == 0))
    {
        // This is an ENQUEUE notification.
        return (0);
    }
    if ((event & LBM_EVENT_QUEUE_SIZE_WARNING) != 0)
    {
        // Size warning, event queue size is in evq_size
    }
    if ((event & LBM_EVENT_QUEUE_DELAY_WARNING) != 0)
    {
        // Delay warning, delay of oldest (about to be dequeued)
        // message is in event_delay_usec.
    }
    return (0);
}
```

**8.1.3.13 `typedef struct lbm_event_queue_stats_t_stct lbm_event_queue_stats_t`**

This structure holds statistics for messages and other events that enter and exit the event queue. NOTE: Specific count-enable options must sometimes be enabled for these statistics to populate.

**8.1.3.14 `typedef int(*) lbm_fd_cb_proc(lbm_context_t *ctx, lbm_handle_t handle, lbm_ulong_t ev, void *clientd)`**

Set by **lbg\_register\_fd()**. If this application callback is set without an event queue, it is called from the context thread and is limited in the API calls that it can make.

**Parameters:**

*ctx* Context monitoring the *handle*.

*handle* File descriptor or socket generating the event.

*ev* One or more of LBM\_FD\_EVENT\_\*(ORed together) indicating the event type.

*clientd* Client data pointer supplied in [lbm\\_register\\_fd\(\)](#).

**Returns:**

0 always.

**8.1.3.15 `typedef int(*) lbm_flight_size_set_inflight_cb_proc(int inflight, void *clientd)`**

Set by [lbt\\_\\*\\_flight\\_size\\_set\\_inflight\(\)](#).

**Parameters:**

*inflight* Gives the current inflight value.

*clientd* Client data pointer supplied in the call to [lbt\\_\\*\\_flight\\_size\\_set\\_inflight\(\)](#).

**Returns:**

The new inflight value.

**8.1.3.16 `typedef void(*) lbm_flight_size_set_inflight_ex_cb_proc(lbm_flight_size_inflight_t *inflight, void *clientd)`**

**Parameters:**

*inflight* Pointer to a structure containing current inflight values for both messages and bytes.

*clientd* Client data pointer supplied in the call to [lbt\\_ume\\_flight\\_size\\_set\\_inflight\\_ex\(\)](#).

**8.1.3.17 `typedef int(*) lbm_immediate_msg_cb_proc(lbm_context_t *ctx, lbm_msg_t *msg, void *clientd)`**

Set by [lbt\\_context\\_rcv\\_immediate\\_msgs\(\)](#). If this application callback is set without an event queue, it is called from the context thread and is limited in the API calls that it can make.

**Note:**

For received application messages, be aware that UM does not guarantee any alignment of that data.

**Parameters:**

*ctx* Context receiving the message.

*msg* Pointer to received message.

*clientd* Client data pointer supplied in [lbt\\_context\\_rcv\\_immediate\\_msgs\(\)](#).

**Returns:**

0 always.

**8.1.3.18 `typedef struct lbt_iovec_t_stct lbt_iovec_t`**

UM replacement for struct iovec for portability.

**8.1.3.19 `typedef struct lbt_ipv4_address_mask_t_stct  
lbt_ipv4_address_mask_t`**

A structure used with options to set/get specific addresses within a range.

**8.1.3.20 `typedef int(*) lbt_log_cb_proc(int level, const char *message, void  
*clientd)`**

Set by [lbt\\_log\(\)](#). NOTE: this application callback can be made from the context thread, and is therefore limited in the UM API calls it can make.

**Parameters:**

*level* One of LBM\_LOG\_\* indicating severity level. Values can be (in order of decreasing importance):

- LBM\_LOG\_EMERG
- LBM\_LOG\_ALERT
- LBM\_LOG\_CRIT
- LBM\_LOG\_ERR
- LBM\_LOG\_WARNING
- LBM\_LOG\_NOTICE
- LBM\_LOG\_INFO
- LBM\_LOG\_DEBUG

*message* Pointer to error message string.

*clientd* Client data pointer supplied in [lbm\\_log\(\)](#).

**Returns:**

0 always.

#### **8.1.3.21 `typedef struct lbm_mim_unrecloss_func_t_stct` `lbm_mim_unrecloss_func_t`**

A structure used with options to set/get a specific callback function

#### **8.1.3.22 `typedef int(*) lbm_mim_unrecloss_function_cb(const char` `*source_name, lbm_uint_t seqnum, void *clientd)`**

Set by [lbt\\_context\\_attr\\_setopt\(\)](#) with option "mim\_unrecoverable\_loss\_function".  
NOTE: this application callback is always made from the context thread and is therefore limited in the UM API calls it can make.

**See also:**

[lbt\\_mim\\_unrecloss\\_func\\_t](#)

**Parameters:**

*source\_name* Name of the source

*seqnum* Sequence Number that is lost

*clientd* Client data pointer supplied in the `lbm_mim_unrecloss_func_t` passed to [lbt\\_context\\_attr\\_setopt\(\)](#) with the "mim\_unrecoverable\_loss\_function" attribute.

**Returns:**

0 always.

#### **8.1.3.23 `typedef struct lbm_msg_channel_info_t_stct` `lbm_msg_channel_info_t`**

This channel information assigns a channel designator to individual messages. Receivers may use this channel designator to filter messages or direct them to specific callbacks on a per-channel basis.

#### **8.1.3.24 `typedef struct lbm_msg_fragment_info_t_stct` `lbm_msg_fragment_info_t`**

To retrieve the UM-message fragment information held in this structure, it is typically necessary to call [lbt\\_msg\\_retrieve\\_fragment\\_info\(\)](#).

**8.1.3.25 `typedef struct lbt_msg_gateway_info_t_stct lbt_msg_gateway_info_t`**

**Deprecated**

**8.1.3.26 `typedef struct lbt_msg_ume_deregistration_ex_t_stct lbt_msg_ume_deregistration_ex_t`**

A structure used with UM receivers to indicate successful deregistration (extended form).

**8.1.3.27 `typedef struct lbt_msg_ume_registration_complete_ex_t_stct lbt_msg_ume_registration_complete_ex_t`**

A structure used with UM receivers to indicate successful registration to quorum or to all stores involved.

**8.1.3.28 `typedef struct lbt_msg_ume_registration_ex_t_stct lbt_msg_ume_registration_ex_t`**

A structure used with UM receivers to indicate successful registration (extended form).

**8.1.3.29 `typedef struct lbt_msg_ume_registration_t_stct lbt_msg_ume_registration_t`**

A structure used with UMP receivers to indicate successful registration.

**8.1.3.30 `typedef struct lbt_msg_umq_deregistration_complete_ex_t_stct lbt_msg_umq_deregistration_complete_ex_t`**

A struct used with UMQ receivers to indicate successful de-registration from a queue.

**8.1.3.31 `typedef struct lbt_msg_umq_index_assigned_ex_t_stct lbt_msg_umq_index_assigned_ex_t`**

A structure used with UMQ or ULB receivers to indicate the stop of index assignment from all queue instances involved.

```
8.1.3.32 typedef struct lbm_msg_umq_index_-
assignment_eligibility_start_complete_ex_t_stct
lbm_msg_umq_index_assignment_eligibility_start_complete_ex_t
```

A structure used with UMQ or ULB receivers to indicate the start of index assignment from all queue instances involved.

```
8.1.3.33 typedef struct lbm_msg_umq_index_-
assignment_eligibility_stop_complete_ex_t_stct
lbm_msg_umq_index_assignment_eligibility_stop_complete_ex_t
```

A structure used with UMQ or ULB receivers to indicate the stop of index assignment from all queue instances involved.

```
8.1.3.34 typedef struct lbm_msg_umq_index_released_ex_t_stct
lbm_msg_umq_index_released_ex_t
```

A structure used with UMQ or ULB receivers to indicate the stop of index assignment from all queue instances involved.

```
8.1.3.35 typedef struct lbm_msg_umq_registration_complete_ex_t_stct
lbm_msg_umq_registration_complete_ex_t
```

A structure used with UMQ receivers to indicate successful receiver registration to quorum or to all queue instances involved.

```
8.1.3.36 typedef int(*) lbm_rcv_cb_proc(lbm_rcv_t *rcv, lbm_msg_t *msg, void
*clientd)
```

Set by [lmb\\_rcv\\_create\(\)](#). If this application callback is set without an event queue, it is called from the context thread and is limited in the API calls that it can make.

After the callback returns, the message object *msg* is deleted and the application must not refer to it. This behavior can be overridden by calling [lmb\\_msg\\_retain\(\)](#) from the receive callback before it returns. It then becomes the application's responsibility to delete the message object using [lmb\\_msg\\_delete\(\)](#).

**Note:**

For received application messages, be aware that UM does not guarantee any alignment of that data.

**Parameters:**

*rcv* Receiver object generating the event.

*msg* Message object containing the receiver event.

*clientd* Client data pointer supplied in [lbt\\_rcv\\_create\(\)](#).

**Returns:**

0 always.

#### 8.1.3.37 `typedef void*(* lbt_rcv_src_notification_create_function_cb(const char *source_name, void *clientd)`

Set by [lbt\\_rcv\\_topic\\_attr\\_setopt\(\)](#) with option "source\_notification\_function".  
NOTE: this application callback is always made from the context thread and is therefore limited in the UM API calls it can make.

**See also:**

[lbt\\_rcv\\_src\\_notification\\_func\\_t](#)

**Parameters:**

*source\_name* Name of the source

*clientd* Client data pointer supplied in the `lbt_rcv_src_notification_func_t` passed to [lbt\\_context\\_attr\\_setopt\(\)](#) with the "source\_notification\_function" attribute.

**Returns:**

void pointer to be set for all messages to this topic from the specified source.

#### 8.1.3.38 `typedef int(*) lbt_rcv_src_notification_delete_function_cb(const char *source_name, void *clientd, void *source_clientd)`

Set by [lbt\\_rcv\\_topic\\_attr\\_setopt\(\)](#) with option "source\_notification\_function".  
NOTE: this application callback is always made from the context thread and is therefore limited in the UM API calls it can make.

**See also:**

[lbt\\_rcv\\_src\\_notification\\_func\\_t](#)

**Parameters:**

*source\_name* Name of the source

*clientd* Client data pointer supplied in the `lbt_rcv_src_notification_func_t` passed to [lbt\\_context\\_attr\\_setopt\(\)](#) with the "source\_notification\_function" attribute.

*source\_clientd* Client data pointer set to be included in each message.

**Returns:**

0 always

**8.1.3.39 `typedef struct lbm_rcv_src_notification_func_t_stct`  
`lbm_rcv_src_notification_func_t`**

A structure used with options to set/get a specific callback function

**8.1.3.40 `typedef struct lbm_rcv_transport_stats_daemon_t_stct`  
`lbm_rcv_transport_stats_daemon_t`**

This structure holds statistics for receiver transports using the daemon mode. NOTE: daemon mode is deprecated and no longer available; this structure is retained for backward compatibility only.

**8.1.3.41 `typedef struct lbm_rcv_transport_stats_t_stct`  
`lbm_rcv_transport_stats_t`**

This structure holds statistics for all receiver transports. The structure is filled in when statistics for receiver transports are requested.

**8.1.3.42 `typedef int(*) lbm_request_cb_proc(lbm_request_t *req, lbm_msg_t *msg, void *clientd)`**

Set by [lbt\\_send\\_request\(\)](#), [lbt\\_multicast\\_immediate\\_request\(\)](#), [lbt\\_unicast\\_immediate\\_request\(\)](#). If this application callback is set without an event queue, it is called from the context thread and is limited in the API calls that it can make.

**Note:**

For received application messages, be aware that UM does not guarantee any alignment of that data.

**Parameters:**

*req* Request object receiving the response.

*msg* Pointer to received message.

*clientd* Client data pointer supplied in [lbt\\_send\\_request\(\)](#), etc.

**Returns:**

0 always.

### 8.1.3.43 **typedef int(\*) lbt\_src\_cb\_proc(lbt\_src\_t \*src, int event, void \*ed, void \*clientd)**

Set by [lbt\\_src\\_create\(\)](#) and [lbt\\_hf\\_src\\_create\(\)](#). If this application callback is set without an event queue, it is called from the context thread and is limited in the API calls that it can make.

#### Parameters:

*src* Source object generating the event.

*event* One of LBT\_SRC\_EVENT\_\*

*ed* Pointer to event data, content dependent on event type.

- For *event* == LBT\_SRC\_EVENT\_CONNECT (not applicable for LBT-RM), *ed* should be re-cast as a (char \*) and points at the receiver as a string. Format depends on transport type. Formats containing IP address and Port pertain to the receiver's IP and Port. For string formats and examples, see [lbt\\_transport\\_source\\_info\\_t\\_stct](#).
- For *event* == LBT\_SRC\_EVENT\_DISCONNECT (not applicable for LBT-RM), *ed* should be re-cast as a (char \*) and points at the receiver as a string (see above).
- For *event* == LBT\_SRC\_EVENT\_WAKEUP, *ed* should be re-cast as a (lbt\_src\_event\_wakeup\_t) and indicates which source has become unblocked.
- For *event* == LBT\_SRC\_EVENT\_SEQUENCE\_NUMBER\_INFO, *ed* should be re-cast as a (lbt\_src\_event\_sequence\_number\_info\_t \*) to extract the sequence number information.
- For *event* == LBT\_SRC\_EVENT\_UME\_REGISTRATION\_ERROR, *ed* should be re-cast as a (const char \*) to extract the error message.
- For *event* == LBT\_SRC\_EVENT\_UME\_REGISTRATION\_SUCCESS, *ed* should be re-cast as a (lbt\_src\_event\_ume\_registration\_t \*) to extract the registration information.
- For *event* == LBT\_SRC\_EVENT\_UME\_REGISTRATION\_SUCCESS\_EX, *ed* should be re-cast as a (lbt\_src\_event\_ume\_registration\_ex\_t \*) to extract the extra registration information.
- For *event* == LBT\_SRC\_EVENT\_UME\_REGISTRATION\_COMPLETE\_EX, *ed* should be re-cast as a (lbt\_src\_event\_ume\_registration\_complete\_ex\_t \*) to extract the extra registration completion information.
- For *event* == LBT\_SRC\_EVENT\_UME\_MESSAGE\_STABLE, *ed* should be re-cast as a (lbt\_src\_event\_ume\_ack\_info\_t \*) to extract the UMP message acknowledgment information.
- For *event* == LBT\_SRC\_EVENT\_UME\_MESSAGE\_STABLE\_EX, *ed* should be re-cast as a (lbt\_src\_event\_ume\_ack\_ex\_info\_t \*) to extract the extra UMP message acknowledgment information.

- For *event* == LBM\_SRC\_EVENT\_UME\_DELIVERY\_CONFIRMATION, *ed* should be re-cast as a (*lmb\_src\_event\_ume\_ack\_info\_t* \*) to extract the UMP message acknowledgment information.
- For *event* == LBM\_SRC\_EVENT\_UME\_DELIVERY\_CONFIRMATION\_EX, *ed* should be re-cast as a (*lmb\_src\_event\_ume\_ack\_ex\_info\_t* \*) to extract the extra UMP message acknowledgment information.
- For *event* == LBM\_SRC\_EVENT\_UME\_MESSAGE\_RECLAIMED, *ed* should be re-cast as a (*lmb\_src\_event\_ume\_ack\_info\_t* \*) to extract the UMP message acknowledgment information.
- For *event* == LBM\_SRC\_EVENT\_UME\_STORE\_UNRESPONSIVE, *ed* should be re-cast as a (const char \*) to extract the UMP store name.
- For *event* == LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION, *ed* should be re-cast as a (*lmb\_src\_event\_flight\_size\_notification\_t* \*) to extract the flight size information.
- For *event* == LBM\_SRC\_EVENT\_UME\_MESSAGE\_RECLAIMED\_EX, *ed* should be re-cast as a (*lmb\_src\_event\_ume\_ack\_ex\_info\_t* \*) to extract the UMP message acknowledgment information.
- For all other event types, *ed* contains nothing and should be ignored.

*clientd* Client data pointer supplied in [lmb\\_src\\_create\(\)](#), etc.

#### Returns:

0 always

#### 8.1.3.44 `typedef lmb_uint32_t(*) lmb_src_cost_function_cb(const char *topic, const lmb_transport_source_info_t *transport, lmb_uint32_t hop_count, lmb_uint32_t cost, void *clientd)`

Set via the "source\_cost\_evaluation\_function" context attribute.

#### Parameters:

*topic* Topic for which the new source was discovered.

*transport* Pointer to a [lmb\\_transport\\_source\\_info\\_t](#), describing the transport session.

*hop\_count* Current hop count for the transport session.

*cost* Current cumulative cost for the transport session.

*clientd* Client data pointer supplied when setting "source\_cost\_evaluation\_function" context attribute.

**Returns:**

Application-determined cost for this source as an unsigned 32-bit number. To permanently reject this source, return [LBM\\_SRC\\_COST\\_FUNCTION\\_REJECT](#).

**8.1.3.45 `typedef struct lbt_src_event_flight_size_notification_t_stct`  
`lbt_src_event_flight_size_notification_t`**

A structure used to indicate a state change in flight size status

**8.1.3.46 `typedef struct lbt_src_event_sequence_number_info_t_stct`  
`lbt_src_event_sequence_number_info_t`**

A structure used with UM sources that informs the application the sequence numbers used with a message.

**See also:**

[lbt\\_src\\_send\\_ex](#)

**8.1.3.47 `typedef struct lbt_src_event_ume_ack_ex_info_t_stct`  
`lbt_src_event_ume_ack_ex_info_t`**

A structure used with UMP sources to indicate message acknowledgment by the store and UMP receivers (extended form).

**8.1.3.48 `typedef struct lbt_src_event_ume_ack_info_t_stct`  
`lbt_src_event_ume_ack_info_t`**

A structure used with UMP sources to indicate message acknowledgment by the store and UMP receivers.

**8.1.3.49 `typedef struct lbt_src_event_ume_deregistration_ex_t_stct`  
`lbt_src_event_ume_deregistration_ex_t`**

A structure used with UMP sources to indicate successful deregistration (extended form).

**8.1.3.50 `typedef struct lbm_src_event_ume_registration_complete_ex_t_stct`**  
**`lbm_src_event_ume_registration_complete_ex_t`**

A structure used with UMP sources to indicate successful registration to quorum or to all stores involved.

**8.1.3.51 `typedef struct lbm_src_event_ume_registration_ex_t_stct`**  
**`lbm_src_event_ume_registration_ex_t`**

A structure used with UMP sources to indicate successful registration (extended form).

**8.1.3.52 `typedef struct lbm_src_event_ume_registration_t_stct`**  
**`lbm_src_event_ume_registration_t`**

A structure used with UMP sources to indicate successful registration.

**8.1.3.53 `typedef struct lbm_src_event_umq_message_id_info_t_stct`**  
**`lbm_src_event_umq_message_id_info_t`****See also:**

[lbm\\_src\\_send\\_ex](#) A structure used with UMQ sending applications that informs the application of the UMQ Message ID used with a message.

**8.1.3.54 `typedef struct lbm_src_event_umq_registration_complete_ex_t_stct`**  
**`lbm_src_event_umq_registration_complete_ex_t`**

A structure used with UMQ sources to indicate successful source registration to quorum or to all queue instances involved.

**8.1.3.55 `typedef struct lbm_src_event_umq_stability_ack_info_ex_t_stct`**  
**`lbm_src_event_umq_stability_ack_info_ex_t`**

A structure used with UMQ source applications to indicate message acknowledgment by a queue instance.

**8.1.3.56 `typedef struct lbm_src_event_umq_ulb_message_info_ex_t_stct`**  
**`lbm_src_event_umq_ulb_message_info_ex_t`**

A structure used with UMQ ULB source applications to indicate message events.

**8.1.3.57 `typedef struct lbt_src_event_umq_ulb_receiver_info_ex_t_stct lbt_src_event_umq_ulb_receiver_info_ex_t`**

A structure used with UMQ ULB source applications to indicate receiver events.

**8.1.3.58 `typedef struct lbt_src_event_wakeup_t_stct lbt_src_event_wakeup_t`**

A structure used to indicate the type of source that is now unblocked.

**8.1.3.59 `typedef struct lbt_src_notify_func_t_stct lbt_src_notify_func_t`**

A structure used with options to set/get a specific callback information

**8.1.3.60 `typedef int(*) lbt_src_notify_function_cb(const char *topic_str, const char *src_str, void *clientd)`**

Set by [lbt\\_context\\_attr\\_setopt\(\)](#) with option "resolver\_source\_notification\_function".  
NOTE: this application callback is always made from the context thread, and is therefore limited in the UM API calls it can make.

**See also:**

[lbt\\_src\\_notify\\_func\\_t](#)

**Parameters:**

*topic\_str* Name of topic for which a source has been found.

*src\_str* Source as a string. Format depends on transport type. For string formats and examples, see [lbt\\_transport\\_source\\_info\\_t\\_stct](#).

*clientd* Client data pointer supplied in the `lbt_src_notify_func_t` passed to the [lbt\\_context\\_attr\\_setopt\(\)](#).

**Returns:**

0 always.

**8.1.3.61 `typedef struct lbt_src_send_ex_info_t_stct lbt_src_send_ex_info_t`****See also:**

[lbt\\_src\\_send\\_ex](#)

---

**8.1.3.62 `typedef struct lbm_src_transport_stats_daemon_t_stct  
lbm_src_transport_stats_daemon_t`**

This structure holds statistics for source transports using the daemon mode. NOTE: daemon mode is deprecated and no longer available; this structure is retained for backward compatibility only.

**8.1.3.63 `typedef struct lbm_src_transport_stats_t_stct  
lbm_src_transport_stats_t`**

This structure holds statistics for all source transports. The structure is filled in when statistics for source transports are requested.

**8.1.3.64 `typedef struct lbm_str_hash_func_ex_t_stct lbm_str_hash_func_ex_t`**

A structure used with options to set/get a specific hash function information.

**8.1.3.65 `typedef lbm_ulong_t(*) lbm_str_hash_function_cb(const char *str)`**

Set by [lbtm\\_context\\_attr\\_setopt\(\)](#) with option "resolver\_string\_hash\_function". NOTE: this application callback is always made from the context thread, and is therefore limited in the UM API calls it can make.

**Parameters:**

*str* String to be hashed.

**Returns:**

hash value 0..(lbm\_ulong\_t)-1

**8.1.3.66 `typedef lbm_ulong_t(*) lbm_str_hash_function_cb_ex(const char *str,  
size_t strlen, void *clientd)`**

Set by [lbtm\\_context\\_attr\\_setopt\(\)](#) with option "resolver\_string\_hash\_function\_ex". NOTE: this application callback is always made from the context thread, and is therefore limited in the UM API calls it can make.

**Parameters:**

*str* String to be hashed.

*strlen* Length of str IF AVAILABLE, (lbm\_ulong\_t)-1 if not calculated by lbm

*clientd* Client data pointer supplied in in the lbm\_str\_hash\_func\_ex\_t passed to the [lbtm\\_context\\_attr\\_setopt\(\)](#).

**Returns:**

hash value 0..(lbt\_ulong\_t)-1

**8.1.3.67 `typedef int(*) lbt_timer_cb_proc(lbt_context_t *ctx, const void *clientd)`**

Set by [lbt\\_schedule\\_timer\(\)](#). If this application callback is set without an event queue, it is called from the context thread and is limited in the API calls that it can make.

**Parameters:**

*ctx* Context running the timer.

*clientd* Client data pointer supplied in [lbt\\_schedule\\_timer\(\)](#).

**Returns:**

0 always.

**8.1.3.68 `typedef struct lbt_timeval_t_stct lbt_timeval_t`**

A structure included in UM messages to indicate when the message was received by UM. A message timestamp using this can be up to 500 milliseconds prior to actual receipt time, and hence, is not suitable when accurate message-arrival-time measurements are needed.

**8.1.3.69 `typedef struct lbt_transport_source_info_t_stct lbt_transport_source_info_t`**

This structure holds the fields used to format and/or parse transport source strings. The format of these strings depends mainly on the transport type, as shown below.

- TCP:src\_ip:src\_port[topic\_idx]

example: TCP : 192.168.0.4 : 45789 [1539853954]

- LBTRM:src\_ip:src\_port:session\_id:mc\_group:dest\_port[topic\_idx]

example: LBTRM : 10.29.3.88 : 14390 : e0679abb : 231.13.13.13 : 14400 [1539853954]

- LBT-RU:src\_ip:src\_port:session\_id[topic\_idx] (session\_id optional, per configuration option `transport_lbtru_use_session_id`)

example: LBT-RU : 192.168.3.189 : 34678 [1539853954]

- LBT-IPC:session\_id:transport\_id[topic\_idx]  
example: LBT-IPC:6481f8d4:20000[1539853954]
- LBT-RDMA:src\_ip:src\_port:session\_id[topic\_idx]  
example: LBT-RDMA:192.168.3.189:34678:6471e9c4[1539853954]

Please note that the topic index field (topic\_idx) may or may not be present depending on your version of UM and/or the setting for configuration option source\_includes\_topic\_index.

**See also:**

[lbm\\_transport\\_source\\_format](#) [lbm\\_transport\\_source\\_parse](#)

#### 8.1.3.70 **typedef struct lbm\_unicast\_resolver\_entry\_t\_stct lbm\_unicast\_resolver\_entry\_t**

A structure used with options to get/set information about unicast resolver daemons.

#### 8.1.3.71 **typedef void\*(\* lbm\_ume\_ctx\_rcv\_ctx\_notification\_create\_- function\_cb(const ume\_liveness\_receiving\_context\_t \*rcv, void \*clientd)**

Set by [lbt\\_context\\_attr\\_setopt\(\)](#) with option "lbt\_context\_attr\_ume\_receiver\_-liveness\_notify\_func". NOTE: this application callback is always made from the context thread and is therefore limited in the UM API calls it can make.

**See also:**

[lbt\\_ume\\_rcv\\_ctx\\_notification\\_func\\_t](#)

**Parameters:**

*const* struct ume\_liveness\_receiving\_context\_t

*clientd* Client data pointer supplied in the lbt\_ume\_rcv\_ctx\_notification\_func\_t passed to [lbt\\_context\\_attr\\_setopt\(\)](#) with the "ume\_receiver\_context\_-detection\_function" attribute.

**Returns:**

void pointer to be set for the "unresponsive" event when this ume\_liveness\_-receiving\_context\_t is declared unresponsive.

**8.1.3.72 `typedef int(*) lbm_ume_ctx_rev_ctx_notification_delete_function_cb(const ume_liveness_receiving_context_t *rcv, void *clientd, void *source_clientd)`**

Set by [lbm\\_context\\_attr\\_setopt\(\)](#) with option "lbm\_context\_attr\_ume\_receiver\_liveness\_notify\_func". NOTE: this application callback is always made from the context thread and is therefore limited in the UM API calls it can make.

**See also:**

[lbm\\_ume\\_ctx\\_rev\\_ctx\\_notification\\_func\\_t](#)

**Parameters:**

*const* struct lbm\_ume\_liveness\_rcv\_context\_t

*clientd* Client data pointer supplied in the lbm\_ume\_ctx\_rev\_ctx\_notification\_func\_t passed to [lbm\\_context\\_attr\\_setopt\(\)](#) with the "ume\_receiver\_context\_deletion\_function" attribute.

**Returns:**

0 if success -1 if failure.

**8.1.3.73 `typedef struct lbm_ume_ctx_rev_ctx_notification_func_t_stct lbm_ume_ctx_rev_ctx_notification_func_t`**

A Structure used with options to set/get a specific callback function

**8.1.3.74 `typedef struct lbm_ume_rcv_recovery_info_ex_func_info_t_stct lbm_ume_rcv_recovery_info_ex_func_info_t`**

A structure used with UMP receiver recovery sequence number information callbacks to pass in information as well as return low sequence number information.

**See also:**

[lbm\\_ume\\_rcv\\_recovery\\_info\\_ex\\_func\\_t](#)

**8.1.3.75 `typedef struct lbm_ume_rcv_recovery_info_ex_func_t_stct lbm_ume_rcv_recovery_info_ex_func_t`**

A struct used with options to set/get a specific callback function

---

**8.1.3.76 `typedef int(*) lbm_ume_rcv_recovery_info_ex_function_cb(lbm_ume_rcv_recovery_info_ex_func_info_t *info, void *clientd)`**

Set by [lmb\\_rcv\\_topic\\_attr\\_setopt\(\)](#) with option "ume\_recovery\_sequence\_number\_info\_function". NOTE: this application callback is always made from the context thread, and is therefore limited in the UM API calls it can make.

**See also:**

[lmb\\_ume\\_rcv\\_recovery\\_info\\_ex\\_func\\_t](#)

**Parameters:**

*info* Structure to hold recovery sequence number information in an extended form

*clientd* Client data pointer supplied in the `lbm_ume_rcv_recovery_info_ex_func_t` passed to [lmb\\_rcv\\_topic\\_attr\\_setopt\(\)](#) with the "ume\_recovery\_sequence\_number\_info\_function" attribute.

**Returns:**

0 always

---

**8.1.3.77 `typedef struct lbm_ume_rcv_regid_ex_func_info_t_stct lbm_ume_rcv_regid_ex_func_info_t`**

A structure used with UMP receiver registration ID callbacks to pass in information.

**See also:**

[lmb\\_ume\\_rcv\\_regid\\_func\\_t](#)

---

**8.1.3.78 `typedef struct lbm_ume_rcv_regid_ex_func_t_stct lbm_ume_rcv_regid_ex_func_t`**

A structure used with options to set/get a specific callback function

---

**8.1.3.79 `typedef lbm_uint_t(*) lbm_ume_rcv_regid_ex_function_cb(lbm_ume_rcv_regid_ex_func_info_t *info, void *clientd)`**

Set by [lmb\\_rcv\\_topic\\_attr\\_setopt\(\)](#) with option "ume\_registration\_extended\_function". NOTE: this application callback is always made from the context thread, and is therefore limited in the UM API calls it can make.

See also:

[lbm\\_ume\\_rcv\\_regid\\_ex\\_func\\_t](#)

Parameters:

*info* Structure holding registration information in an extended form

*clientd* Client data pointer supplied in the [lbm\\_ume\\_rcv\\_regid\\_ex\\_func\\_t](#) passed to [lbm\\_rcv\\_topic\\_attr\\_setopt\(\)](#) with the "ume\_registration\_extended\_function" attribute.

Returns:

Registration ID to be used by receiver for given source and topic.

#### 8.1.3.80 `typedef struct lbm_ume_rcv_regid_func_t_stct lbm_ume_rcv_regid_func_t`

A structure used with options to set/get a specific callback function

#### 8.1.3.81 `typedef lbm_uint_t(*) lbm_ume_rcv_regid_function_cb(const char *src_str, lbm_uint_t src_regid, void *clientd)`

Set by [lbm\\_rcv\\_topic\\_attr\\_setopt\(\)](#) with option "ume\_registration\_function". NOTE: this application callback is always made from the context thread, and is therefore limited in the UM API calls it can make.

See also:

[lbm\\_ume\\_rcv\\_regid\\_func\\_t](#)

Parameters:

*src\_str* Name of the source for the ID.

*src\_regid* Registration ID for the source for this topic.

*clientd* Client data pointer supplied in the [lbm\\_ume\\_rcv\\_regid\\_func\\_t](#) passed to [lbm\\_rcv\\_topic\\_attr\\_setopt\(\)](#) with the "ume\_registration\_function" attribute.

Returns:

Registration ID to be used by receiver for given source and topic.

#### 8.1.3.82 `typedef struct lbm_ume_src_force_reclaim_func_t_stct lbm_ume_src_force_reclaim_func_t`

A structure used with options to set/get a specific callback function

**8.1.3.83 `typedef int(*) lbm_ume_src_force_reclaim_function_cb(const char *topic_str, lbm_uint_t seqnum, void *clientd)`**

Set by [lbm\\_src\\_topic\\_attr\\_setopt\(\)](#) with option "ume\_force\_reclaim\_function".  
NOTE: this application callback is always made from the context thread and is therefore limited in the UM API calls it can make.

**See also:**

[lbm\\_ume\\_src\\_force\\_reclaim\\_func\\_t](#)

**Parameters:**

*topic\_str* Name of the topic for the reclaim

*seqnum* Sequence Number that is reclaimed

*clientd* Client data pointer supplied in the `lbm_ume_src_force_reclaim_func_t` passed to [lbm\\_src\\_topic\\_attr\\_setopt\(\)](#) with the "ume\_force\_reclaim\_function" attribute.

**Returns:**

0 always.

**8.1.3.84 `typedef struct lbm_ume_store_entry_t_stct lbm_ume_store_entry_t`**

A structure used with options to get/set information for a UMP store

**8.1.3.85 `typedef struct lbm_ume_store_group_entry_t_stct lbm_ume_store_group_entry_t`**

A structure used with options to get/set information for a UMP store group

**8.1.3.86 `typedef struct lbm_ume_store_name_entry_t_stct lbm_ume_store_name_entry_t`**

A structure used with options to get/set information for a UMP store

**8.1.3.87 `typedef struct lbm_umq_index_info_t_stct lbm_umq_index_info_t`**

A structure used with UM sources and receivers to associated UMQ Indices with messages.

**8.1.3.88 `typedef struct lbt_umq_msg_total_lifetime_info_t_stct lbt_umq_msg_total_lifetime_info_t`**

A structure used with UMQ sources to specify a message's total lifetime.

**8.1.3.89 `typedef struct lbt_umqmsgid_t_stct lbt_umqmsgid_t`**

See also:

[lbt\\_umq\\_regid\\_t](#) A structure used with UMQ messages to identify a message uniquely.

**8.1.3.90 `typedef struct lbt_umq_queue_entry_t_stct lbt_umq_queue_entry_t`**

A struct used with options to get/set Registration ID information for UMQ queues

**8.1.3.91 `typedef lbt_uint64_t lbt_umq_regid_t`**

Registration ID used for UMQ contexts for both sources and receivers

**8.1.3.92 `typedef struct lbt_umq_ulb_application_set_attr_t_stct lbt_umq_ulb_application_set_attr_t`**

A struct used with options to get/set UMQ ULB application set attributes

**8.1.3.93 `typedef struct lbt_umq_ulb_receiver_type_attr_t_stct lbt_umq_ulb_receiver_type_attr_t`**

A struct used with options to get/set UMQ ULB receiver type attributes

**8.1.3.94 `typedef struct lbt_umq_ulb_receiver_type_entry_t_stct lbt_umq_ulb_receiver_type_entry_t`**

A struct used with options to get/set UMQ ULB Receiver Type entries

**8.1.3.95 `typedef struct lbt_wildcard_rcv_compare_func_t_stct lbt_wildcard_rcv_compare_func_t`**

A structure used with options to set/get a specific application callback pattern type.

---

**8.1.3.96** `typedef int(*) lbm_wildcard_rcv_compare_function_cb(const char *topic_str, void *clientd)`

Set by [lbm\\_wildcard\\_rcv\\_attr\\_setopt\(\)](#) with option "pattern\_callback". NOTE: this application callback is always made from the context thread, and is therefore limited in the UM API calls it can make.

**See also:**

[lbm\\_wildcard\\_rcv\\_compare\\_func\\_t](#)

**Parameters:**

*topic\_str* Name of topic to be checked for match.

*clientd* Client data pointer supplied in the `lbm_wildcard_rcv_compare_func_t` passed to [lbm\\_wildcard\\_rcv\\_attr\\_setopt\(\)](#) with the "pattern\_callback" attribute.

**Returns:**

0 for match and 1 for no match.

---

**8.1.3.97** `typedef struct lbm_wildcard_rcv_create_func_t_stct lbm_wildcard_rcv_create_func_t`

A structure used with options to set/get a specific wildcard topic receiver creation callback type.

---

**8.1.3.98** `typedef int(*) lbm_wildcard_rcv_create_function_cb(const char *topic_str, lbm_rcv_topic_attr_t *attr, void *clientd)`

Set by [lbm\\_wildcard\\_rcv\\_attr\\_setopt\(\)](#) with option "receiver\_create\_callback". NOTE: this application callback is always made from the context thread, and is therefore limited in the UM API calls it can make.

**See also:**

[lbm\\_wildcard\\_rcv\\_create\\_func\\_t](#)

**Parameters:**

*topic\_str* Name of topic which was matched, and for which a receiver will be created.

*attr* Pointer to an `lbm_rcv_topic_attr_t` which has been initialized with the receiver options which will be used to create the receiver.

***clientd*** Client data pointer supplied in the `lbt_wildcard_rcv_create_func_t` passed to `lbt_wildcard_rcv_attr_setopt()` with the "receiver\_create\_callback" attribute.

**Returns:**

Always return 0.

**8.1.3.99 `typedef struct lbt_wildcard_rcv_delete_func_t_stct lbt_wildcard_rcv_delete_func_t`**

A structure used with options to set/get a specific wildcard topic receiver deletion callback type.

**8.1.3.100 `typedef int(*) lbt_wildcard_rcv_delete_function_cb(const char *topic_str, void *clientd)`**

Set by `lbt_wildcard_rcv_attr_setopt()` with option "receiver\_delete\_callback". NOTE: this application callback is always made from the context thread, and is therefore limited in the UM API calls it can make.

**See also:**

[lbt\\_wildcard\\_rcv\\_delete\\_func\\_t](#)

**Parameters:**

***topic\_str*** Name of topic which was matched, and for which a receiver will be deleted.

***clientd*** Client data pointer supplied in the `lbt_wildcard_rcv_delete_func_t` passed to `lbt_wildcard_rcv_attr_setopt()` with the "receiver\_delete\_callback" attribute.

**Returns:**

Always return 0.

**8.1.3.101 `typedef struct ume_liveness_receiving_context_t_stct ume_liveness_receiving_context_t`**

A structure used to hold a receiving context's user rcv regid and session id. Source contexts use this information to track receiver liveness.

### 8.1.4 Function Documentation

**8.1.4.1 LBMEExpDLL int lbm\_apphdr\_chain\_append\_elem  
(lbm\_apphdr\_chain\_t \* *chain*, lbm\_apphdr\_chain\_elem\_t \* *elem*)**

**Parameters:**

*chain* Pointer to an app header chain.

*elem* Pointer to a user-created app header element.

**Returns:**

0 for success, -1 for failure

**8.1.4.2 LBMEExpDLL int lbm\_apphdr\_chain\_create (lbm\_apphdr\_chain\_t \*\*  
*chain*)**

**Parameters:**

*chain* Pointer to a pointer to an app header chain. This will be filled in with the newly created chain.

**Returns:**

0 for success, -1 for failure

**8.1.4.3 LBMEExpDLL int lbm\_apphdr\_chain\_delete (lbm\_apphdr\_chain\_t \*  
*chain*)**

**Parameters:**

*chain* Pointer to an app header chain.

**Returns:**

0 for success, -1 for failure

**8.1.4.4 LBMEExpDLL int lbm\_apphdr\_chain\_iter\_create  
(lbm\_apphdr\_chain\_iter\_t \*\* *chain\_iter*, lbm\_apphdr\_chain\_t \* *chain*)**

**Parameters:**

*chain\_iter* Pointer to a pointer to an lbm\_apphdr\_chain\_iter\_t structure to be filled in.

*chain* Pointer to an app header chain from which to create the iterator.

**Returns:**

0 if the iterator points to the first element in the chain, -1 if there are no elements in the chain

**8.1.4.5 LBMEExpDLL int lbm\_apphdr\_chain\_iter\_create\_from\_msg  
(lbm\_apphdr\_chain\_iter\_t \*\**chain\_iter*, lbm\_msg\_t \**msg*)****Parameters:**

*chain\_iter* Pointer to a pointer to an lbm\_apphdr\_chain\_elem\_t structure to be filled in

*msg* Pointer to a UM message from which to retrieve the app header chain.

**Returns:**

0 if the iterator points to the first element in the chain, -1 if there are no elements in the chain

**8.1.4.6 LBMEExpDLL lbm\_apphdr\_chain\_elem\_t\* lbm\_-  
apphdr\_chain\_iter\_current (lbm\_apphdr\_chain\_iter\_t \*\**chain\_iter*)****Parameters:**

*chain\_iter* Pointer to pointer to an lbm\_apphdr\_chain\_iter\_t iterator.

**Returns:**

lbm\_apphdr\_chain\_elem\_t pointer to the current app header chain element.

**8.1.4.7 LBMEExpDLL int lbm\_apphdr\_chain\_iter\_delete  
(lbm\_apphdr\_chain\_iter\_t \**chain\_iter*)****Parameters:**

*chain\_iter* Pointer to an lbm\_apphdr\_chain\_iter\_t created by one of the lbm\_- apphdr\_chain\_iter\_create functions.

**Returns:**

0 for success, -1 for failure

**8.1.4.8 LBMExpDLL int lbm\_apphdr\_chain\_iter\_done  
(lbm\_apphdr\_chain\_iter\_t \*\* *chain\_iter*)****Parameters:***chain\_iter* Pointer to pointer to an lbm\_apphdr\_chain\_iter\_t iterator.**Returns:**

1 if there is a next element in an app header chain, 0 otherwise.

**8.1.4.9 LBMExpDLL int lbm\_apphdr\_chain\_iter\_first  
(lbm\_apphdr\_chain\_iter\_t \*\* *chain\_iter*)****Parameters:***chain\_iter* Pointer to pointer to an lbm\_apphdr\_chain\_iter\_t iterator.**Returns:**

0 for Success and -1 for Failure.

**8.1.4.10 LBMExpDLL int lbm\_apphdr\_chain\_iter\_next  
(lbm\_apphdr\_chain\_iter\_t \*\* *chain\_iter*)****Parameters:***chain\_iter* Pointer to pointer to an lbm\_apphdr\_chain\_iter\_t iterator.**Returns:**

0 for Success, -1 for failure if there is no next element in the chain (iterator is unmodified).

**8.1.4.11 LBMExpDLL int lbm\_async\_operation\_cancel  
(lbm\_async\_operation\_handle\_t *handle*, int *flags*)**

Calling this function will cause the associated asynchronous operation's async operation callback function to be called with a canceled status. If the operation could not be canceled (either it has already completed, it never existed, or it is currently executing and past the point of no return), then -1 is returned and [lbm\\_errnum\(\)](#) is set to indicate why the operation could not be canceled. Otherwise, 0 is returned for a successful cancel, indicating that the operation was found and guaranteed to have been truly canceled.

**Warning:**

It is generally not safe to call this function from within an asynchronous operation callback for the same handle that is being canceled. There is one exception: it is safe to call cancel on a handle from within the initial LBM\_ASYNC\_OP\_STATUS\_IN\_PROGRESS that delivers the handle; this is in fact a reasonable way to simulate a non-blocking synchronous call.

Once an operation has been canceled, any associated [lbtm\\_async\\_operation\\_info\\_t](#) objects are no longer valid and should not be accessed. This includes access to the opinfo parameter from within an initial LBM\_ASYNC\_OP\_STATUS\_IN\_PROGRESS callback at any point in that callback after cancel has been called.

**Parameters:**

*handle* Handle to the asynchronous operation.

*flags* Flags to affect the behavior of the cancel. ORed set of values.

- LBM\_ASYNC\_OPERATION\_CANCEL\_FLAG\_NONBLOCK - If operation cannot be canceled immediately, return without canceling. The default behavior is to block until the operation can be successfully canceled.

**Returns:**

-1 for Failure or 0 for Success.

**8.1.4.12 LBMEExpDLL int lbtm\_async\_operation\_status  
(lbtm\_async\_operation\_handle\_t *handle*, int *flags*)**

Calling this function will cause the associated asynchronous operation's async operation callback function to be called with current status information. This is a merely a polling mechanism, and the information returned is guaranteed to be correct only for the duration of the async operation callback function. It may change immediately afterwards.

**Warning:**

It is not safe to call this function from within an asynchronous operation callback for the same handle that status is being requested for.

**Parameters:**

*handle* Handle to the asynchronous operation.

*flags* Flags to affect the behavior of the status request. ORed set of values.

- LBM\_ASYNC\_OPERATION\_STATUS\_FLAG\_NONBLOCK - If the operation's status cannot be retrieved immediately, just return without blocking. The default behavior is to block until the operation's status can be retrieved.

**Returns:**

-1 for Failure or 0 for Success.

**8.1.4.13 LBMEExpDLL int lbm\_auth\_set\_credentials (lbm\_context\_t \* ctx, const char \* name, size\_t name\_len, const char \* passwd, size\_t passwd\_len, lbm\_cred\_callback\_fn cbfn, void \* clientd, int auth\_required)**

Calling this function will set the credential of the user and make the requirement for the authentication. There are two ways to set credential: either setting the user's name and password parameters or passing the callback function pointer to retrieve credential information. The callback function method will override the credentials set by the input parameters. Once the parameter of auth\_required is set to "1", the authentication results will be examined and errors will be reported if authentication checks fail. If the "auth\_required" is set to "0", then the authentication failure will be ignored and there is no impact on the undergoing process.

**Parameters:**

*ctx* LBM context object.  
*name* the user name string.  
*name\_len* the length of the user name string.  
*passwd* the user password string.  
*passwd\_len* the length of the user password string.  
*cbfn* the callback function pointer.  
*clientd* the parameter of the callback function.  
*auth\_required* the variable to require authentication service

**Returns:**

-1 for Failure or 0 for Success.

**8.1.4.14 LBMEExpDLL int lbm\_authstorage\_addtpnam (const char \* username, const char \* pass, unsigned char flags)**

Calling this function will generate new credential entry for the user and save it to the password file. Setting parameter of "flags" to "1" will overwrite the existing entry for the same user.

**Parameters:**

*username* the user's name string.

*pass* the password string.

*flags* overwritting flag.

**Returns:**

negative values for Failure or 0 and positive values for Success.

#### **8.1.4.15 LBMExpDLL int lbtm\_authstorage\_checkpermission (char \* username, char \* command)**

Calling this function will check if the user is authorized to execute the specified command.

**Parameters:**

*username* the user's name string.

*command* the command string.

**Returns:**

-1 for Failure or 0 for Denial or 1 for Success.

#### **8.1.4.16 LBMExpDLL void lbtm\_authstorage\_close\_storage\_xml (void)**

Calling this function will release the storage object created by [lbtm\\_authstorage\\_open\\_storage\\_xml\(\)](#).

**Parameters:**

*None.*

**Returns:**

None.

#### **8.1.4.17 LBMExpDLL int lbtm\_authstorage\_deltpnam (const char \* username)**

Calling this function will remove the credential entry for the user from the password file.

**Parameters:**

*username* the user's name string.

**Returns:**

-1 for Failure or 0 for Success.

**8.1.4.18 LBMExpDLL int lbm\_authstorage\_load\_roletable ()**

Calling this function will create an internal data object to hold the role table from the password file.

**Parameters:**

*None.*

**Returns:**

-1 for Failure or 0 for Success.

**8.1.4.19 LBMExpDLL int lbm\_authstorage\_open\_storage\_xml (char \* *filename*)**

Calling this function will create the storage object which contains all users' authentication and authorization information from the XML password file with the name specified in the input parameter. If that file does not exist, a default password information will be used instead.

**Parameters:**

*filename* the xml file name string.

**Returns:**

0 for Success or negative for failure (-1:invalid parameter; -2: storage exist; -3: creation failed)

**8.1.4.20 LBMExpDLL int lbm\_authstorage\_print\_roletable ()**

Calling this function will print out the role table saved in the internal data object created by [lbm\\_authstorage\\_load\\_roletable\(\)](#) function.

**Parameters:**

*None.*

**Returns:**

-1 for Failure or 0 for Success.

**8.1.4.21 LBMEExpDLL int lbm\_authstorage\_roletable\_add\_role\_action (const char \* rolename, const char \* action)**

Calling this function will authorize users assuming the specified role to perform the assigned action.

**Parameters:**

*rolename* the role name string.

*action* the action name string

**Returns:**

-1 for Failure or 0 for Success.

**8.1.4.22 LBMEExpDLL int lbm\_authstorage\_unload\_roletable ()**

Calling this function will release the role table saved in the internal data object.

**Parameters:**

*None.*

**Returns:**

-1 for Failure or 0 for Success.

**8.1.4.23 LBMEExpDLL int lbm\_authstorage\_user\_add\_role (const char \* username, const char \* role)**

Calling this function will add a new role entry for the specified user to the password file.

**Parameters:**

*username* the user's name string.

*role* the role string.

**Returns:**

-1 for Failure or 0 for Success.

**8.1.4.24 LBMExpDLL int lbm\_authstorage\_user\_del\_role (const char \*  
*username*, const char \**role*)**

Calling this function will remove the role entry for the specified user from the password file.

**Parameters:**

*username* the user's name string.

*role* the role string.

**Returns:**

-1 for Failure or 0 for Success.

**8.1.4.25 LBMExpDLL int lbm\_cancel\_fd (lbm\_context\_t \**ctx*, lbm\_handle\_t  
*handle*, lbm\_ulong\_t *ev*)**

Cancel a previously registered file descriptor/socket event. Note that there are rare circumstances where this function can return while the fd callback may still be executing. If the application needs to know when all possible processing on the fd is complete, it must use [lbm\\_cancel\\_fd\\_ex\(\)](#).

**See also:**

[lbm\\_register\\_fd](#)

**Warning:**

It is not recommended to call this function from a context thread callback.

**Parameters:**

*ctx* Pointer to the UM context object.

*handle* file descriptor/socket of interest for event.

*ev* One or more of LBM\_FD\_EVENT\_\*(ORed together). Mask of events to cancel.

**Returns:**

0 for Success and -1 for Failure.

#### 8.1.4.26 LBMEExpDLL int lbm\_cancel\_fd\_ex (lbm\_context\_t \* *ctx*, lbm\_handle\_t *handle*, lbm\_ulong\_t *ev*, lbm\_event\_queue\_cancel\_cb\_info\_t \* *cbinfo*)

Cancel a previously registered file descriptor/socket event, with an application callback indicating when the fd is fully canceled. This extended version of the fd cancel function requires the configuration option queue\_cancellation\_callbacks\_enabled be set to 1.

**See also:**

[lbm\\_register\\_fd](#)

**Parameters:**

*ctx* Pointer to the UM context object.

*handle* file descriptor/socket of interest for event.

*ev* Mask of events to cancel.

*cbinfo* Cancellation callback information, containing the event queue, function pointer, and client data for the callback.

**Returns:**

0 for Success and -1 for Failure.

#### 8.1.4.27 LBMEExpDLL int lbm\_cancel\_timer (lbm\_context\_t \* *ctx*, int *id*, void \*\* *clientdp*)

Cancel a previously scheduled timer. The timer is identified by the return value of the [lbm\\_schedule\\_timer\(\)](#) function. If the passed-in timer ID is not valid, this cancel function returns success, which occurs if the passed-in timer ID has already fired or if the timer ID is garbage. Note that there are rare circumstances where this function can return while the timer callback may still be executing. If the application needs to know when all possible processing on the timer is complete, it must use [lbm\\_cancel\\_timer\\_ex\(\)](#).

**See also:**

[lbm\\_schedule\\_timer](#)

**Parameters:**

*ctx* Pointer to the UM context object.

*id* The identifier specifying the timer to cancel

*clientdp* Pointer to a client data pointer. This function sets it to the client data pointer supplied by the [lbm\\_schedule\\_timer\(\)](#). If the caller does not need the client data, it can pass NULL as *clientdp*.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.28 LBMExpDLL int lbm\_cancel\_timer\_ex (lbm\_context\_t \* ctx, int id,  
void \*\* clientdp, lbm\_event\_queue\_cancel\_cb\_info\_t \* cbinfo)**

Cancel a previously scheduled timer, with an application callback indicating when the timer is fully canceled. The timer is identified by the return value of the [lbm\\_schedule\\_timer\(\)](#) function. If the passed-in timer ID is not valid, this cancel function returns success, which occurs if the passed-in timer ID has already fired or if the timer ID is garbage. This extended version of the timer cancel function requires the configuration option queue\_cancellation\_callbacks\_enabled be set to 1.

**See also:**

[lbm\\_schedule\\_timer](#)

**Parameters:**

*ctx* Pointer to the UM context object.

*id* The identifier specifying the timer to cancel

*clientdp* Pointer to a client data pointer. This function sets it to the client data pointer supplied by the [lbm\\_schedule\\_timer\(\)](#). If the caller does not need the client data, it can pass NULL as *clientdp*.

*cbinfo* Cancellation callback information, containing the event queue, function pointer, and client data for the callback.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.29 LBMExpDLL int lbm\_config (const char \* fname)****Parameters:**

*fname* String containing the file name or URL (tftp or http) that contains the options to parse and set. File names with a ".xml" extension will be passed to [lbm\\_config\\_xml\\_file\(\)](#) with a NULL application name.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.30 LBMEExpDLL int lbm\_config\_xml\_file (const char \* *url*, const char \* *application\_name*)**

Parse the xml configuration file specified by url, and apply the configuration for the given application name. UM XML configuration may only be loaded once in the lifetime of a process. If the LBM\_UMM\_INFO or LBM\_XML\_CONFIG\_FILENAME environment variables are set and they are successful in setting UM XML configuration, this API will have no effect and return -1.

**Parameters:**

- url* String containing the path to the XML configuration file. A URL beginning with `http://` or `ftp://` may also be provided.  
*application\_name* The name of this application which must match an application tag in the XML configuration file. This parameter may be NULL, in which case the application tag with no name is matched.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.31 LBMEExpDLL int lbm\_config\_xml\_string (const char \* *xml\_data*, const char \* *application\_name*)**

Parse the xml configuration data contained in *xml\_data*, and apply the configuration for the given application name. UM XML configuration may only be loaded once in the lifetime of a process. If the LBM\_UMM\_INFO or LBM\_XML\_CONFIG\_FILENAME environment variables are set and they are successful in setting UM XML configuration, this API will have no effect and return -1.

**Parameters:**

- xml\_data* String containing UM XML configuration data.  
*application\_name* The name of this application which must match an application tag in the XML configuration data. This parameter may be NULL, in which case the application tag with no name is matched.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.32 LBMEExpDLL int lbm\_context\_attr\_create (lbm\_context\_attr\_t \*\* *attr*)**

The attribute object is allocated and filled with the current default values that are used by *lbm\_context\_t* objects and may have been modified by a previously loaded configuration file.

**Parameters:**

*attr* A pointer to a pointer to a UM context attribute structure. Will be filled in by this function to point to the newly created `lbm_context_attr_t` object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.33 LBMEExpDLL int `lbm_context_attr_create_default`  
(`lbm_context_attr_t **attr`)**

The attribute object is allocated and filled with the initial or factory default values built into LBM that are used by `lbm_context_t` objects.

**Parameters:**

*attr* A pointer to a pointer to a UM context attribute structure. Will be filled in by this function to point to the newly created `lbm_context_attr_t` object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.34 LBMEExpDLL int `lbm_context_attr_create_from_xml`  
(`lbm_context_attr_t **attr, const char *context_name`)**

The attribute object is allocated and filled with the current default values that are used by `lbm_context_t` objects and may have been modified by a previously loaded configuration file. Then, if an XML configuration file has been loaded, the attribute object is further filled with the defaults for the given context name. If the context name is not permitted by the XML configuration, -1 is returned and no attribute object is created.

**Parameters:**

*attr* A pointer to a pointer to a UM context attribute structure. Will be filled in by this function to point to the newly created `lbm_context_attr_t` object.

*context\_name* The context name used to lookup this context in the XML configuration. A NULL value is permitted, and will match unnamed contexts defined in the XML. The context name is also written into the attribute object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.35 LBMEExpDLL int lbm\_context\_attr\_delete (lbm\_context\_attr\_t \* attr)**

The attribute object is cleaned up and deleted.

**Parameters:**

*attr* Pointer to a UM context attribute object as returned by [lbm\\_context\\_attr\\_create](#).

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.36 LBMEExpDLL int lbm\_context\_attr\_dump (lbm\_context\_attr\_t \* cattr, int \* size, lbm\_config\_option\_t \* opts)**

The config object is filled with context configuration options

**Parameters:**

*cattr* The context attribute object to retrieve the attributes from

*size* Size of the opts array. Will return the number of items that were set in opts

*opts* The options array to fill

**8.1.4.37 LBMEExpDLL int lbm\_context\_attr\_dup (lbm\_context\_attr\_t \*\* attr, const lbm\_context\_attr\_t \* original)**

A new attribute object is created as a copy of an existing object.

**Parameters:**

*attr* A pointer to a pointer to a UM context attribute structure. Will be filled in by this function to point to the newly created lbm\_context\_attr\_t object.

*original* Pointer to a UM context attribute object as returned by [lbm\\_context\\_attr\\_create](#) or [lbm\\_context\\_attr\\_create\\_default](#), from which *attr* is initialized.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.38 LBMExpDLL int lbm\_context\_attr\_getopt (lbm\_context\_attr\_t \*attr,  
const char \* optname, void \* optval, size\_t \* optlen)****Parameters:**

*attr* Pointer to a UM context attributed object.

*optname* String containing the option name.

*optval* Pointer to the option value structure to be filled. The structure of the option values are specific to the options themselves.

*optlen* Length (in bytes) of the *optval* structure when passed in. Upon return, this is set to the size of the optval filled in structure.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.39 LBMExpDLL int lbm\_context\_attr\_option\_size ()**

The function returns the number of entries that are of type "context"

**Returns:**

The number of entries that are of type "context"

**8.1.4.40 LBMExpDLL int lbm\_context\_attr\_set\_from\_xml (lbm\_context\_attr\_t  
\* attr, const char \* context\_name)**

The attribute object is filled with the default values for the given context name, if an XML configuration file has been loaded. If the context name is not permitted by the XML configuration, -1 is returned and no values are set.

**Parameters:**

*attr* A pointer to a UM context attribute structure.

*context\_name* The context name used to lookup this context in the XML configuration. A NULL value is permitted, and will match unnamed contexts defined in the XML. The context name is also written into the attribute object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.41 LBMEExpDLL int lbm\_context\_attr\_setopt (lbm\_context\_attr\_t \* attr,  
const char \* optname, const void \* optval, size\_t optlen)**

Used before the context is created. NOTE: the attribute object must first be initialized with the corresponding \_attr\_create() function.

**Parameters:**

- attr** Pointer to a UM context attribute object.
- optname** String containing the option name.
- optval** Pointer to the option value structure. The structure of the option values are specific to the options themselves.
- optlen** Length (in bytes) of the *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.42 LBMEExpDLL int lbm\_context\_attr\_str\_getopt (lbm\_context\_attr\_t \*  
attr, const char \* optname, char \* optval, size\_t \* optlen)****Parameters:**

- attr** Pointer to a UM context attributed object.
- optname** String containing the option name.
- optval** Pointer to the string to be filled in.
- optlen** Maximum length (in bytes) of the *string* when passed in. Upon return, this is set to the size of the formatted string.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.43 LBMEExpDLL int lbm\_context\_attr\_str\_setopt (lbm\_context\_attr\_t \*  
attr, const char \* optname, const char \* optval)**

Used before the context is created. NOTE: the attribute object must first be initialized with the corresponding \_attr\_create() function.

**Parameters:**

- attr** Pointer to a UM context attributed object.
- optname** String containing the option name.

*optval* String containing the option value. The format of the string is specific to the option itself.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.44 LBMExpDLL int lbm\_context\_create (lbm\_context\_t \*\* *ctxp*, const lbm\_context\_attr\_t \* *attr*, lbm\_daemon\_event\_cb\_proc *proc*, void \* *clientd*)**

This creates an instance of the UM main processing element, a UM context. Sources and Receivers are created from a UM context and work within that context. For the Embedded operational mode, a thread is spawned to handle event processing. For Sequential operational mode, the application "donates" an execution thread by calling [lbm\\_context\\_process\\_events\(\)](#).

**See also:**

[lbm\\_context\\_delete\(\)](#)

**Parameters:**

*ctxp* A pointer to a pointer to a UM context object. Will be filled in by this function to point to the newly created `lbm_context_t` object.

*attr* A pointer to a UM context attribute object. A value of NULL will use default attributes.

*proc* A callback function to call when events occur on the UM daemon connection. NOTE: daemon mode is no longer available; this parameter is retained for backward compatibility only. Please pass NULL.

*clientd* Client data to pass into the UM daemon event callback. NOTE: daemon mode is no longer available; this parameter is retained for backward compatibility only. Please pass NULL.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.45 LBMExpDLL int lbm\_context\_delete (lbm\_context\_t \* *ctx*)**

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*ctx* Pointer to a UM context object to delete.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.46 LBMEExpDLL int lbtm\_context\_delete\_ex (lbtm\_context\_t \* *ctx*,  
lbtm\_event\_queue\_cancel\_cb\_info\_t \* *cbinfo*)****Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*ctx* Pointer to a UM context object to delete.

*cbinfo* Cancellation callback information, containing the event queue, function pointer, and client data for the callback.

**Returns:**

0 for Success and -1 for Failure

**8.1.4.47 LBMEExpDLL int lbtm\_context\_dump (lbtm\_context\_t \* *ctx*, int \* *size*,  
lbtm\_config\_option\_t \* *opts*)**

The config object is filled with context configuration options

**Parameters:**

*ctx* The context object to retrieve the attributes from

*size* Size of the *opts* array. Will return the number of items that were set in *opts*

*opts* The options array to fill

**8.1.4.48 LBMEExpDLL lbtm\_context\_t\* lbtm\_context\_from\_rcv (lbtm\_rev\_t \*  
*rcv*)****Parameters:**

*rcv* Pointer to a UM receiver object.

**Returns:**

A pointer to the UM context object associated with the UM receiver object.

**8.1.4.49 LBMExpDLL *lbm\_context\_t\** *lbm\_context\_from\_src* (*lbm\_src\_t \* src*)****Parameters:**

*src* Pointer to a UM source object.

**Returns:**

A pointer to the UM context object associated with the UM source object.

**8.1.4.50 LBMExpDLL *lbm\_context\_t\** *lbm\_context\_from\_wildcard\_rcv* (*lbm\_wildcard\_rcv\_t \* wcrcv*)****Parameters:**

*wcrcv* Pointer to a UM wildcard receiver object.

**Returns:**

A pointer to the LBM context object associated with the LBM wildcard receiver object.

**8.1.4.51 LBMExpDLL int *lbm\_context\_get\_name* (*lbm\_context\_t \* ctx*, *char \* name*, *size\_t \* size*)****Parameters:**

*ctx* Pointer to an existing UM context object.

*name* Pointer to a buffer into which is stored the context name.

*size* Pointer to a variable holding the size of the buffer. If the buffer is not large enough, this will be filled in with the required size.

**8.1.4.52 LBMExpDLL int *lbm\_context\_getopt* (*lbm\_context\_t \* ctx*, *const char \* optname*, *void \* optval*, *size\_t \* optlen*)****Parameters:**

*ctx* Pointer to a UM context object where the option is stored.

*optname* String containing the option name.

*optval* Pointer to the option value structure. The structure of the option values are specific to the options themselves.

*optlen* When passed, this is the max length (in bytes) of the *optval* structure.  
When returned, this is the length of the filled in *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.53 LBMEExpDLL int lbm\_context\_lbtipc\_unblock (lbm\_context\_t \* ctx)**

When transport\_lbtipc\_receiver\_operational\_mode is set to LBM\_CTX\_ATTR\_OP\_SEQUENTIAL (or "sequential"), then it is the responsibility of the application to explicitly process LBT-IPC messages for the UM context. This function allows an application to cause [lbm\\_context\\_process\\_lbtipc\\_messages\(\)](#) to immediately return instead of continuing to process messages.

**Parameters:**

*ctx* Pointer to the UM context object.

**8.1.4.54 LBMEExpDLL int lbm\_context\_process\_events (lbm\_context\_t \* ctx, lbm\_ulong\_t msec)**

When opmode is set to LBM\_CTX\_ATTR\_OP\_SEQUENTIAL (or "sequential"), then it is the responsibility of the application to explicitly process events for the UM context. This function will process timers and file descriptor/socket events for internal processing as well as API timer and file descriptor/socket events. The application thread that is processing events must remain active until the context is deleted.

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*ctx* Pointer to the UM context object.

*msec* Continue event processing loop for at least *msec* milliseconds before returning.

**Returns:**

0 for Success and -1 for Failure.

**Note:**

It is the responsibility of the application to "unblock" this function using "lbm\_context\_unblock()" and cease further calls before deleting the UM context.

#### 8.1.4.55 LBMEExpDLL int lbm\_context\_process\_lbtipc\_messages (lbm\_context\_t \* *ctx*, lbm\_ulong\_t *msec*, lbm\_ulong\_t *loop\_count*)

When `transport_lbtipc_receiver_operational_mode` is set to `LBM_CTX_ATTR_OP_SEQUENTIAL` (or "sequential"), then it is the responsibility of the application to explicitly process LBT-IPC messages for the UM context. This function will satisfy that requirement.

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*ctx* Pointer to the UM context object.

*msec* Only used if `transport_lbtipc_receiver_thread_behavior` is set to "pend".

The timeout in milliseconds of the pend waiting for new data (actual Operating System resolution may vary). Defaults to no timeout on Operating Systems that do not support a timeout (e.g. Mac OS X). A value of zero will result in "busy\_wait" like behavior on all Operating Systems.

*loop\_count* Number of loops before returning whether or not data has been received. Zero results in looping forever.

**Returns:**

0 for Success and -1 for Failure.

**Note:**

It is the responsibility of the application to "unblock" this function using [`lbm\_context\_lbtipc\_unblock\(\)`](#) and cease further calls before deleting the UM context.

#### 8.1.4.56 LBMEExpDLL int lbm\_context\_rcv\_immediate\_msgs (lbm\_context\_t \* *ctx*, lbm\_immediate\_msg\_cb\_proc *proc*, void \* *clientd*, lbm\_event\_queue\_t \* *evq*)

**Parameters:**

*ctx* Pointer to a UM context object that listens for messages.

*proc* Pointer to a function to call when a message arrives.

*clientd* Client data passed when a message is delivered.

*evq* Optional Event Queue to place messages on when they arrive. If NULL causes *proc* to be called from context thread.

**8.1.4.57 LBMEExpDLL int lbm\_context\_rcv\_immediate\_topic\_msgs  
(lbm\_context\_t \* *ctx*, lbm\_immediate\_msg\_cb\_proc *proc*, void \*  
*clientd*, lbm\_event\_queue\_t \* *evq*)**

**Parameters:**

- ctx* Pointer to a UM context object that listens for messages.
- proc* Pointer to a function to call when a message arrives.
- clientd* Client data passed when a message is delivered.
- evq* Optional Event Queue to place messages on when they arrive. If NULL causes *proc* to be called from context thread.

**8.1.4.58 LBMEExpDLL int lbm\_context\_reactor\_only\_create (lbm\_context\_t \*\*  
*ctxp*, const lbm\_context\_attr\_t \* *attr*)**

This creates an instance of the UM main processing element, a UM context. However, this version of the context is only usable for timer and file descriptor event handling. It can not be used for source or receiver creation, etc. For the Embedded operational mode, a thread is spawned to handle event processing. For Sequential operational mode, the application "donates" an execution thread by calling [lbm\\_context\\_process\\_events\(\)](#).

**See also:**

[lbm\\_context\\_create](#)

**Parameters:**

- ctxp* A pointer to a pointer to a UM context object. Will be filled in by this function to point to the newly created lbm\_context\_t object.
- attr* A pointer to a UM context attribute object. A value of NULL will use default attributes.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.59 LBMEExpDLL int lbm\_context\_reset\_im\_rcv\_transport\_stats  
(lbm\_context\_t \* *ctx*)**

**Parameters:**

- ctx* Pointer to the UM context to reset statistics for.
- stats* Pointer to a stats structure to fill in.

**Returns:**

-1 for Failure and 0 for Success.

**8.1.4.60 LBMExpDLL int lbm\_context\_reset\_im\_src\_transport\_stats  
(lbm\_context\_t \* ctx)****Parameters:**

*ctx* Pointer to the UM context to reset statistics for.

*stats* Pointer to a stats structure to fill in.

**Returns:**

-1 for Failure and 0 for Success.

**8.1.4.61 LBMExpDLL int lbm\_context\_reset\_rcv\_transport\_stats  
(lbm\_context\_t \* ctx)****Parameters:**

*ctx* Pointer to the UM context to reset statistics for.

**Returns:**

-1 for Failure and 0 for Success.

**8.1.4.62 LBMExpDLL int lbm\_context\_reset\_src\_transport\_stats  
(lbm\_context\_t \* ctx)****Parameters:**

*ctx* Pointer to the UM context to reset statistics for.

**Returns:**

-1 for Failure and 0 for Success.

**8.1.4.63 LBMExpDLL int lbm\_context\_reset\_stats (lbm\_context\_t \* ctx)****Parameters:**

*ctx* Pointer to the UM context to reset statistics for.

**Returns:**

-1 for Failure and 0 for Success.

**8.1.4.64 LBMEExpDLL int lbm\_context\_retrieve\_im\_recv\_transport\_stats  
(lbm\_context\_t \* *ctx*, int \* *num*, int *size*, lbm\_recv\_transport\_stats\_t \*  
*stats*)**

**Parameters:**

***ctx*** Pointer to the UM context to retrieve statistics for.

***num*** Pointer to an integer that must hold the maximum number of elements in the stats array when passed in. Upon return, this value is set to the number of sources filled in.

***size*** Size in bytes of each entry in *stats*

***stats*** Array of lbm\_recv\_transport\_stats\_t objects to fill in transport stats for.

**Returns:**

-1 for Failure and 0 for Success.

**Note:**

If -1 is returned, and [lbt\\_errnum\(\)](#) returns LBM\_EINVAL, then \**num* may contain a larger number than the value originally passed into this function. This return value represents the number of lbm\_recv\_transport\_stats\_t objects required in the *stats* array and can be used to dynamically determine how many entries are needed.

**8.1.4.65 LBMEExpDLL int lbm\_context\_retrieve\_im\_src\_transport\_stats  
(lbm\_context\_t \* *ctx*, int \* *num*, int *size*, lbm\_src\_transport\_stats\_t \*  
*stats*)**

**Parameters:**

***ctx*** Pointer to the UM context to retrieve statistics for.

***num*** Pointer to an integer that must hold the maximum number of elements in the stats array when passed in. Upon return, this value is set to the number of sources filled in.

***size*** Size in bytes of each entry in *stats*

***stats*** Array of lbm\_src\_transport\_stats\_t objects to fill in transport stats for.

**Returns:**

-1 for Failure and 0 for Success.

**Note:**

If -1 is returned, and [lbt\\_errnum\(\)](#) returns LBM\_EINVAL, then \**num* may contain a larger number than the value originally passed into this function. This return value represents the number of lbm\_src\_transport\_stats\_t objects required in the *stats* array and can be used to dynamically determine how many entries are needed.

**8.1.4.66 LBMExpDLL int lbm\_context\_retrieve\_rcv\_transport\_stats  
(lbm\_context\_t \* *ctx*, int \* *num*, lbm\_rcv\_transport\_stats\_t \* *stats*)****Parameters:**

*ctx* Pointer to the UM context to retrieve statistics for.

*num* Pointer to an integer that must hold the maximum number of elements in the stats array when passed in. Upon return, this value is set to the number of sources filled in.

*stats* Array of lbm\_rcv\_transport\_stats\_t objects to fill in transport stats for.

**Returns:**

-1 for Failure and 0 for Success.

**Note:**

If -1 is returned, and [lbt\\_errnum\(\)](#) returns LBM\_EINVAL, then \**num* may contain a larger number than the value originally passed into this function. This return value represents the number of lbm\_rcv\_transport\_stats\_t objects required in the *stats* array and can be used to dynamically determine how many entries are needed.

**8.1.4.67 LBMExpDLL int lbm\_context\_retrieve\_src\_transport\_stats  
(lbm\_context\_t \* *ctx*, int \* *num*, lbm\_src\_transport\_stats\_t \* *stats*)****Parameters:**

*ctx* Pointer to the UM context to retrieve statistics for.

*num* Pointer to an integer that must hold the maximum number of elements in the stats array when passed in. Upon return, this value is set to the number of sources filled in.

*stats* Array of lbm\_src\_transport\_stats\_t objects to fill in transport stats for.

**Returns:**

-1 for Failure and 0 for Success.

**Note:**

If -1 is returned, and [lbt\\_errnum\(\)](#) returns LBM\_EINVAL, then \**num* may contain a larger number than the value originally passed into this function. This return value represents the number of lbm\_src\_transport\_stats\_t objects required in the *stats* array and can be used to dynamically determine how many entries are needed.

**8.1.4.68 LBMEExpDLL int lbm\_context\_retrieve\_stats (lbm\_context\_t \* *ctx*,  
                  lbm\_context\_stats\_t \* *stats*)****Parameters:**

*ctx* Pointer to the UM context to retrieve statistics for.

*stats* Pointer to a stats structure to fill in.

**Returns:**

-1 for Failure and 0 for Success.

**8.1.4.69 LBMEExpDLL int lbm\_context\_set\_name (lbm\_context\_t \* *ctx*, const  
                  char \* *name*)****Parameters:**

*ctx* Pointer to an existing UM context object.

*name* The context name. Context names are limited in length to 128 characters (not including the final null) and restricted to alphanumeric characters, hyphens, and underscores.

**8.1.4.70 LBMEExpDLL int lbm\_context\_setopt (lbm\_context\_t \* *ctx*, const char  
                  \* *optname*, const void \* *optval*, size\_t *optlen*)**

Only those options that can be set during operation may be specified. See "Options That May Be Set During Operation" (doc/Config/maybesetduringoperation.html) in The UM Configuration Guide. For API functions that can access any option, see *lbm\_context\_attr\_\**().

**Parameters:**

*ctx* Pointer to a UM context object where the option is to be set.

*optname* String containing the option name.

*optval* Pointer to the option value structure. The structure of the option values are specific to the options themselves.

*optlen* Length (in bytes) of the *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.71 LBMExpDLL int lbm\_context\_str\_getopt (lbm\_context\_t \* *ctx*, const char \* *optname*, char \* *optval*, size\_t \* *optlen*)****Parameters:**

*ctx* Pointer to a UM context object where the option is stored.  
*optname* String containing the option name.  
*optval* String to hold the option value.  
*optlen* When passed, this is the max length (in bytes) of the string. When returned, this is the length of the filled in option value.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.72 LBMExpDLL int lbm\_context\_str\_setopt (lbm\_context\_t \* *ctx*, const char \* *optname*, const char \* *optval*)**

Only those options that can be set during operation may be specified. See "Options That May Be Set During Operation" (doc/Config/maybesetduringoperation.html) in The UM Configuration Guide. For API functions that can access any option, see *lbm\_context\_attr\_\**().

**Parameters:**

*ctx* Pointer to a UM context object where the option is to be set.  
*optname* String containing the option name.  
*optval* String containing the option value. The format of the string is specific to the options themselves.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.73 LBMExpDLL int lbm\_context\_topic\_resolution\_request  
(lbm\_context\_t \* *ctx*, lbm\_ushort\_t *flags*, lbm\_ulong\_t *interval\_msec*,  
lbm\_ulong\_t *duration\_sec*)****Parameters:**

*ctx* Pointer to a UM context object.  
*flags* Flags indicating desired requests. ORed set of values.

- LBM\_TOPIC\_RES\_REQUEST\_ADVERTISEMENT - Request advertisements from quiescent sources.

- LBM\_TOPIC\_RES\_REQUEST\_QUERY - Request queries from quiescent receivers.
- LBM\_TOPIC\_RES\_REQUEST\_WILDCARD\_QUERY - Request queries from quiescent wildcard receivers.

*interval\_msec* Interval between requests in milliseconds. Less than 10 should be used with caution. Less than 5 is not recommended.

*duration\_sec* Minimum duration of requests in seconds. Actual duration can be longer depending upon the interval. A value of zero will result in 1 request and the interval will be meaningless.

**Returns:**

0 for Success and -1 for Failure

#### 8.1.4.74 LBMEExpDLL int lbm\_context\_unblock (lbm\_context\_t \* *ctx*)

When opmode is set to LBM\_CTX\_ATTR\_OP\_SEQUENTIAL (or "sequential"), then it is the responsibility of the application to explicitly process events for the UM context. This function allows an application to cause *lbm\_process\_events()* to immediately return instead of continuing to process events.

**Parameters:**

*ctx* Pointer to the UM context object.

#### 8.1.4.75 LBMEExpDLL lbm\_uint64\_t lbm\_create\_random\_id ()

**Returns:**

a random 64 bit long.

#### 8.1.4.76 LBMEExpDLL int lbm\_ctx\_umq\_get\_inflight (lbm\_context\_t \* *ctx*, const char \* *qname*, int \* *inflight*, lbm\_flight\_size\_set\_inflight\_cb\_proc *proc*, void \* *clientd*)

**See also:**

[lbm\\_flight\\_size\\_set\\_inflight\\_cb\\_proc](#)

**Parameters:**

*ctx* Pointer to the context.

*qname* Name of the queue.

*inflight* Pointer to an int whose value will be filled in to reflect the current inflight.

*proc* Optional callback that allows an application to set the current inflight.

*clientid* Optional client data passed into the proc.

**Returns:**

0 for Success, -1 for failure if the proc returns a negative value.

**8.1.4.77 LBMExpDLL int lbm\_ctx\_umq\_queue\_topic\_list (lbm\_context\_t \*  
                  ctx, const char \* queue\_name, lbm\_async\_operation\_func\_t \*  
                  async\_opfunc)**

The returned list of topics is complete once the asynchronous operation callback is called with an LBM\_ASYNC\_OP\_STATUS\_COMPLETE. Each returned lbm\_umq\_queue\_topic\_t object also contains the application sets associated with that topic and receiver type IDs associated with each application set.

**See also:**

[lbm\\_umq\\_queue\\_topic\\_t](#)

**Parameters:**

*ctx* LBM context object.

*queue\_name* Name of the queue to retrieve a topic list from.

*async\_opfunc* The asynchronous operation callback the topic list will be delivered to.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.78 LBMExpDLL int lbm\_debug\_dump (const char \*filename, int append)**

**Parameters:**

*filename* to open and dump debug log events to

*append* Flag to indicate that the dump should be appended to the file or overwrite the file

**8.1.4.79 LBMEExpDLL void lbm\_debug\_filename (const char \*filename)****Warning:**

May be overridden by environment variable

**Parameters:**

*filename* to open and send log events to

**8.1.4.80 LBMEExpDLL void lbm\_debug\_mask (lbm\_uint64\_t mask)****Warning:**

May be overridden by environment variable

**Parameters:**

*mask* of debug log events to log (contact support for more information)

**8.1.4.81 LBMEExpDLL lbm\_response\_t\* lbm\_deserialize\_response  
(lbm\_context\_t \*ctx, lbm\_serialized\_response\_t \*serialized\_response)**

De-serializes a serialized UM response object, making it usable for [lbm\\_send\\_response\(\)](#). Note that the returned *lbm\_response\_t* object should be treated as any other normal response object, and deleted by the application using [lbm\\_response\\_delete\(\)](#) as appropriate.

**Parameters:**

*ctx* A pointer to a UM context object.

*serialized\_response* A pointer to a serialized UM response object.

**Returns:**

A pointer to a *lbm\_response\_t* object.

**8.1.4.82 LBMEExpDLL const char\* lbm\_errmsg (void)****Returns:**

Pointer to a static char array holding the error message.

**8.1.4.83 LBMExpDLL int lbm\_errnum (void)****Returns:**

Integer error number.

**8.1.4.84 LBMExpDLL int lbm\_event\_dispatch (lbm\_event\_queue\_t \* evq,  
lbm\_ulong\_t tmo)****Parameters:**

*evq* Event Queue that holds the events to dispatch.

*tmo* The number of milliseconds to block before returning from the function. Note that if no events are posted, the call will continue to block even after the time has past. See <https://communities.informatica.com/infakb/faq/5/Pages/80007.aspx> for details. In addition to numeric values, the following special values are valid:

- LBM\_EVENT\_QUEUE\_BLOCK - block indefinitely processing events.
- LBM\_EVENT\_QUEUE\_POLL - poll and dispatch a single event and return.

**Returns:**

> 0 for Success (number returned is the number of events serviced) or -1 for Failure.

**8.1.4.85 LBMExpDLL int lbm\_event\_dispatch\_unblock (lbm\_event\_queue\_t \*  
evq)**

This function enqueues a special event into the event queue that, when processed, causes the thread calling `lbm_event_dispatch` to return.

**Parameters:**

*evq* Event Queue on which to enqueue the UNBLOCK event.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.86 LBMEExpDLL int lbt\_event\_queue\_attr\_create  
(lbt\_event\_queue\_attr\_t \*\* attr)**

The attribute object is allocated and filled with the current default values that are used by lbt\_event\_queue\_t objects and may have been modified by a previously loaded configuration file.

**Parameters:**

*attr* A pointer to a pointer to a UM event queue attribute structure. Will be filled in by this function to point to the newly created lbt\_event\_queue\_attr\_t object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.87 LBMEExpDLL int lbt\_event\_queue\_attr\_create\_default  
(lbt\_event\_queue\_attr\_t \*\* attr)**

The attribute object is allocated and filled with the initial or factory default values built into LBM that are used by lbt\_event\_queue\_t objects that concern receivers.

**Parameters:**

*attr* A pointer to a pointer to a UM event queue attribute structure. Will be filled in by this function to point to the newly created lbt\_event\_queue\_attr\_t object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.88 LBMEExpDLL int lbt\_event\_queue\_attr\_create\_from\_xml  
(lbt\_event\_queue\_attr\_t \*\* attr, const char \* event\_queue\_name)**

The attribute object is allocated and filled with the current default values that are used by lbt\_event\_queue\_t objects and may have been modified by a previously loaded configuration file. Then, if an XML configuration file has been loaded, the attribute object is further filled with the defaults for the given event queue name. If the event queue name is not permitted by the XML configuration, -1 is returned and no attribute object is created.

**Parameters:**

*attr* A pointer to a pointer to a UM event queue attribute structure. Will be filled in by this function to point to the newly created lbt\_event\_queue\_attr\_t object.

***event\_queue\_name*** The event queue name used to lookup this event queue in the XML configuration. A NULL value is permitted, and will match unnamed event queues defined in the XML. The event queue name is also written into the attribute object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.89 LBMEExpDLL int lbm\_event\_queue\_attr\_delete  
(lbm\_event\_queue\_attr\_t \* attr)**

The attribute object is cleaned up and deleted.

**Parameters:**

***attr*** Pointer to a UM event queue attribute object as returned by [lbm\\_event\\_queue\\_attr\\_create](#).

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.90 LBMEExpDLL int lbm\_event\_queue\_attr\_dump  
(lbm\_event\_queue\_attr\_t \* eatr, int \* size, lbm\_config\_option\_t \* opts)**

The config object is filled with event queue configuration options

**Parameters:**

***eattr*** The event queue attribute object to retrieve the attributes from

***size*** Size of the opts array. Will return the number of items that were set in opts

***opts*** The options array to fill

**8.1.4.91 LBMEExpDLL int lbm\_event\_queue\_attr\_dup  
(lbm\_event\_queue\_attr\_t \*\* attr, const lbm\_event\_queue\_attr\_t \* original)**

A new attribute object is created as a copy of an existing object.

**Parameters:**

***attr*** A pointer to a pointer to a UM event queue attribute structure. Will be filled in by this function to point to the newly created lbm\_event\_queue\_attr\_t object.

***original*** Pointer to a UM event queue attribute object as returned by [lbt\\_event\\_queue\\_attr\\_create](#) or [lbt\\_event\\_queue\\_attr\\_create\\_default](#), from which *attr* is initialized.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.92 LBMEExpDLL int lbt\_event\_queue\_attr\_getopt  
(lbt\_event\_queue\_attr\_t \* *attr*, const char \* *optname*, void \* *optval*,  
size\_t \* *optlen*)****Parameters:**

***attr*** Pointer to a UM event queue attribute object where the option is stored

***optname*** String containing the option name

***optval*** Pointer to the option value structure. The structure of the option values are specific to the options themselves.

***optlen*** When passed, this is the max length (in bytes) of the *optval* structure.  
When returned, this is the length of the filled in *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.93 LBMEExpDLL int lbt\_event\_queue\_attr\_option\_size ()**

The function returns the number of entries that are of type "event queue"

**Returns:**

The number of entries that are of type "event queue"

**8.1.4.94 LBMEExpDLL int lbt\_event\_queue\_attr\_set\_from\_xml  
(lbt\_event\_queue\_attr\_t \* *attr*, const char \* *event\_queue\_name*)**

The attribute object is filled with the default values for the given event queue name, if an XML configuration file has been loaded. If the event queue name is not permitted by the XML configuration, -1 is returned and no values are set.

**Parameters:**

***attr*** A pointer to a UM event queue attribute structure.

***event\_queue\_name*** The event queue name used to lookup this event queue in the XML configuration. A NULL value is permitted, and will match unnamed event queues defined in the XML. The event queue name is also written into the attribute object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.95 LBMExpDLL int lbm\_event\_queue\_attr\_setopt  
(lbm\_event\_queue\_attr\_t \*attr, const char \*optname, const void \*  
optval, size\_t optlen)**

Used before the event queue is created. NOTE: the attribute object must first be initialized with the corresponding \_attr\_create() function.

**Parameters:**

***attr*** Pointer to a UM event queue attribute object where the option is to be set  
***optname*** String containing the option name  
***optval*** Pointer to the option value structure. The structure of the option values are specific to the options themselves.  
***optlen*** Length (in bytes) of the *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.96 LBMExpDLL int lbm\_event\_queue\_attr\_str\_getopt  
(lbm\_event\_queue\_attr\_t \*attr, const char \*optname, char \*optval,  
size\_t \*optlen)**

**Parameters:**

***attr*** Pointer to a UM event queue attribute object where the option is stored  
***optname*** String containing the option name  
***optval*** String to be filled in with the option value.  
***optlen*** When passed, this is the max length (in bytes) of the string. When returned, this is the length of the filled in option value.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.97 LBMEExpDLL int lbm\_event\_queue\_attr\_str\_setopt  
(lbm\_event\_queue\_attr\_t \* attr, const char \* optname, const char \*  
optval)**

Used before the event queue is created. NOTE: the attribute object must first be initialized with the corresponding \_attr\_create() function.

**Parameters:**

*attr* Pointer to a UM event queue attribute object where the option is to be set  
*optname* String containing the option name  
*optval* String containing the option value. The format of the string is specific to the options themselves.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.98 LBMEExpDLL int lbm\_event\_queue\_create (lbm\_event\_queue\_t \*\*  
evqp, lbm\_event\_queue\_monitor\_proc proc, void \* clientd, const  
lbm\_event\_queue\_attr\_t \* attr)**

This function creates an event queue that may be passed in several functions in order for events/callbacks to be queued for execution.

**Parameters:**

*evqp* A pointer to a pointer for the lbm\_event\_queue\_t object created to be stored.  
*proc* Pointer to function to call when monitoring the event queue.  
*clientd* Client data returned in the callback proc *proc*.  
*attr* A pointer to an event queue attribute object or NULL for default attributes.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.99 LBMEExpDLL int lbm\_event\_queue\_delete (lbm\_event\_queue\_t \* evq)**

**Warning:**

An event queue should not be deleted before all other dependent objects (source, receivers, and timers using the event queue) have also been deleted or canceled.

**Parameters:**

*evq* Event Queue to be deleted.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.100 LBMEpdLL int lbm\_event\_queue\_dump (lbm\_event\_queue\_t \* *evq*,  
int \* *size*, lbm\_config\_option\_t \* *opts*)**

The config object is filled with event queue configuration options

**Parameters:**

*evq* The event queue object to retrieve the attributes from

*size* Size of the opts array. Will return the number of items that were set in opts

*opts* The options array to fill

**8.1.4.101 LBMEpdLL lbm\_event\_queue\_t\* lbm\_event\_queue\_from\_rcv  
(lbm\_rcv\_t \* *rcv*)****Parameters:**

*rcv* Pointer to a UM receiver object.

**Returns:**

A pointer to the UM event queue object associated with the UM receiver object.

**8.1.4.102 LBMEpdLL lbm\_event\_queue\_t\* lbm\_event\_queue\_from\_src  
(lbm\_src\_t \* *src*)****Parameters:**

*src* Pointer to a UM source object.

**Returns:**

A pointer to the UM event queue object associated with the UM source object.

**8.1.4.103 LBMEExpDLL *lbt\_event\_queue\_t*\* *lbt\_event\_queue\_from\_wildcard\_recv* (*lbt\_wildcard\_recv\_t* \* *wcrcv*)**

**Parameters:**

*wcrcv* Pointer to a UM wildcard receiver object.

**Returns:**

A pointer to the LBM event queue object associated with the LBM wildcard receiver object.

**8.1.4.104 LBMEExpDLL int *lbt\_event\_queue\_getopt* (*lbt\_event\_queue\_t* \* *evq*, const char \* *optname*, void \* *optval*, size\_t \* *optlen*)**

**Parameters:**

*evq* Pointer to a UM event queue object where the option is stored

*optname* String containing the option name

*optval* Pointer to the option value structure. The structure of the option values are specific to the options themselves.

*optlen* When passed, this is the max length (in bytes) of the *optval* structure.  
When returned, this is the length of the filled in *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.105 LBMEExpDLL int *lbt\_event\_queue\_reset\_stats* (*lbt\_event\_queue\_t* \* *evq*)**

**Parameters:**

*evq* Pointer to the UM event queue to reset statistics for.

**Returns:**

-1 for Failure and 0 for Success.

**8.1.4.106 LBMEExpDLL int *lbt\_event\_queue\_retrieve\_stats* (*lbt\_event\_queue\_t* \* *evq*, *lbt\_event\_queue\_stats\_t* \* *stats*)**

**Parameters:**

*evq* Pointer to the UM event queue to retrieve statistics for.

*stats* Pointer to a stats structure to fill in.

**Returns:**

-1 for Failure and 0 for Success.

**8.1.4.107 LBMEExpDLL int lbm\_event\_queue\_setopt (lbm\_event\_queue\_t \* *evq*, const char \* *optname*, const void \* *optval*, size\_t *optlen*)**

Only those options that can be set during operation may be specified. See "Options That May Be Set During Operation" (doc/Config/maybesetduringoperation.html) in The UM Configuration Guide. For API functions that can access any option, see *lbm\_event\_queue\_attr\_\**().

**Parameters:**

*evq* Pointer to a UM event queue where the option is to be set

*optname* String containing the option name

*optval* Pointer to the option value structure. The structure of the option values are specific to the options themselves.

*optlen* Length (in bytes) of the *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.108 LBMEExpDLL int lbm\_event\_queue\_shutdown (lbm\_event\_queue\_t \* *evq*)**

**Parameters:**

*evq* Event Queue to shutdown.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.109 LBMEExpDLL int lbm\_event\_queue\_size (lbm\_event\_queue\_t \* *evq*)**

This call is only supported when the *queue\_size\_warning* config variable is set. If not set, then this function will return -1 and set an EINVAL error.

**Parameters:**

*evq* Event Queue to determine the size for.

**Returns:**

> 0 indicates the size of the event queue and -1 for Failure.

**8.1.4.110 LBMEExpDLL int lbm\_event\_queue\_str\_getopt (lbm\_event\_queue\_t \*  
evq, const char \* optname, char \* optval, size\_t \* optlen)****Parameters:**

*evq* Pointer to a UM event queue object where the option is stored

*optname* String containing the option name

*optval* String to be filled in with the option value.

*optlen* When passed, this is the max length (in bytes) of the string. When returned, this is the length of the filled in with the option value.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.111 LBMEExpDLL int lbm\_event\_queue\_str\_setopt (lbm\_event\_queue\_t \*  
evq, const char \* optname, const char \* optval)**

Only those options that can be set during operation may be specified. See "Options That May Be Set During Operation" (doc/Config/maybesetduringoperation.html) in The UM Configuration Guide. For API functions that can access any option, see *lbm\_event\_queue\_attr\_\**().

**Parameters:**

*evq* Pointer to a UM event queue object where the option is to be set

*optname* String containing the option name

*optval* String containing the option value. The format of the string is specific to the options themselves.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.112 LBMEExpDLL char\* lbm\_get\_jms\_msg\_id (lbm\_uint64\_t source\_id,  
lbm\_uint64\_t seqno\_id, char \* topic)****Returns:**

a JMS Message ID string.

---

**8.1.4.113 LBMExpDLL int `lbt_hf_rcv_create` (`lbt_hf_rcv_t ** hfrcvp,  
lbt_context_t * ctx, lbt_topic_t * topic, lbt_rcv_cb_proc proc, void  
* clientd, lbt_event_queue_t * evq)`)**

**Warning:**

It is not safe to call this function from a context thread callback.

**See also:**

[lbt\\_rcv\\_create](#)

**Parameters:**

*hfrcvp* A pointer to a pointer to a UM Hot Failover (HF) receiver object. Will be filled in by this function to point to the newly created `lbt_fd_rcv_t` object.

*ctx* Pointer to the LBM context object associated with the sender.

*topic* Pointer to the LBM topic object associated with the desired receiver topic.

**Warning:**

Topic references should not be reused. Each `lbt_hf_rcv_create()` call should be preceded by a call to `lbt_rcv_topic_lookup()`.

**Parameters:**

*proc* Pointer to a function to call when messages arrive.

*clientd* Pointer to client data that is passed when data arrives and *proc* is called.

*evq* Optional Event Queue to place message events on when they arrive. If NULL causes *proc* to be called from context thread.

**Returns:**

0 for Success and -1 for Failure.

---

**8.1.4.114 LBMExpDLL int `lbt_hf_rcv_delete` (`lbt_hf_rcv_t * hfrcv`)**

Delete a UM Hot Failover (HF) receiver object. Note that this function can return while the receivercallback may still be executing if receiver events are being delivered via an event queue. If the application needs to know when all possible processing on the receiver is complete, it must use `lbt_hf_rcv_delete_ex()`.

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*hfrcv* Pointer to a UM HF receiver object to delete.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.115 LBMEExpDLL int lbm\_hf\_rcv\_delete\_ex (lbm\_hf\_rcv\_t \* *hfrcv*,  
                          lbm\_event\_queue\_cancel\_cb\_info\_t \* *cbinfo*)**

Delete a UM Hot Failover (HF) receiver object, with an application callback indicating when the receiver is fully canceled. This extended version of the receiver delete function requires the configuration option `queue_cancellation_callbacks_enabled` be set to 1.

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*hfrcv* Pointer to a UM HF receiver object to delete.

*cbinfo* Cancellation callback information, containing the event queue, function pointer, and client data for the callback.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.116 LBMEExpDLL lbm\_hf\_rcv\_t\* lbm\_hf\_rcv\_from\_rcv (lbm\_rcv\_t \* *rcv*)****Parameters:**

*rcv* Pointer to a UM receiver object.

**Returns:**

Pointer to a UM HF receiver for the receiver object or NULL if none exists.

**8.1.4.117 LBMEExpDLL int lbm\_hf\_rcv\_topic\_dump (lbm\_hf\_rcv\_t \* *hfrcv*, int  
                          \*size, lbm\_config\_option\_t \* *opts*)**

The config object is filled with receiver configuration options

**Parameters:**

*hfrcv* The HF receiver object to retrieve the attributes from  
*size* Size of the opts array. Will return the number of items that were set in opts  
*opts* The options array to fill

**8.1.4.118 LBMEExpDLL int lbm\_hf\_src\_create (lbm\_src\_t \*\**srcp*,  
 lbm\_context\_t \**ctx*, lbm\_topic\_t \**topic*, lbm\_src\_cb\_proc *proc*, void  
 \**clientd*, lbm\_event\_queue\_t \**evq*)**

**See also:**

[lbm\\_src\\_create](#)

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*srcp* A pointer to a pointer to a UM source object. Will be filled in by this function to point to the newly created lbm\_src\_t object.  
*ctx* Pointer to the LBM context object associated with the sender.  
*topic* Pointer to the LBM topic object associated with the destination of messages sent by the source.  
*proc* Pointer to a function to call when events occur related to the source. If NULL, then events are not delivered to the source.  
*clientd* Pointer to client data that is passed when *proc* is called.  
*evq* Optional Event Queue to place events on when they occur. If NULL causes *proc* to be called from context thread.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.119 LBMEExpDLL int lbm\_hf\_src\_send (lbm\_src\_t \**src*, const char \**msg*, size\_t *len*, lbm\_uint\_t *sqn*, int *flags*)**

The LBM source must have been created with lbm\_hf\_src\_create and not lbm\_src\_create

**See also:**

[lbm\\_src\\_send](#)

**Warning:**

It is not recommended to call this function from a context thread callback. If called from a context thread callback, use the LBM\_SRC\_NONBLOCK flag and handle any LBM\_EWOULDBLOCK errors internally.

**Parameters:**

*src* Pointer to the LBM source to send from

*msg* Pointer to the data to send in this message

*len* Length (in bytes) of the data to send in this message

*sqn* The application sequence number to associate with this message.

*flags* Flags indicating various conditions. ORed set of values.

- LBM\_MSG\_START\_BATCH - Message starts a batch of messages
- LBM\_MSG\_END\_BATCH - Message ends a batch of messages. Batch should be sent to the implicit batching buffer.
- LBM\_MSG\_COMPLETE\_BATCH - Message constitutes a complete batch and should be sent to the implicit batching buffer.
- LBM\_MSG\_FLUSH - Message is to be sent ASAP (not implicitly or explicitly batched). This also flushes waiting messages that were explicitly or implicitly batched.
- LBM\_SRC\_NONBLOCK - If message could not be sent immediately return and error and signal LBM\_EWOULDBLOCK.
- LBM\_SRC\_BLOCK - Block the caller indefinitely until the message is sent. (This behavior is the default if neither LBM\_SRC\_NONBLOCK nor LBM\_SRC\_BLOCK are supplied.)

**Returns:**

-1 for Failure or 0 for Success.

**8.1.4.120 LBMDLL int lbm\_hf\_src\_send\_ex (lbm\_src\_t \*src, const char \*msg, size\_t len, lbm\_uint\_t sqn, int flags, lbm\_src\_send\_ex\_info\_t \*exinfo)**

The LBM source must have been created with `lbm_hf_src_create` and not `lbm_src_create`

**See also:**

[lbm\\_src\\_send\\_ex](#)

**Warning:**

If called from a context thread callback, use the LBM\_SRC\_NONBLOCK flag and handle any LBM\_EWOULDBLOCK errors internally.

Calling this function from a context thread callback for stability and confirmation events could cause a deadlock

**Parameters:**

*src* Pointer to the LBM source to send from

*msg* Pointer to the data to send in this message

*len* Length (in bytes) of the data to send in this message

*sqn* The application sequence number to associate with this message.

*flags* Flags indicating various conditions. ORed set of values.

- LBM\_MSG\_START\_BATCH - Message starts a batch of messages
- LBM\_MSG\_END\_BATCH - Message ends a batch of messages. Batch should be sent to the implicit batching buffer.
- LBM\_MSG\_COMPLETE\_BATCH - Message constitutes a complete batch and should be sent to the implicit batching buffer.
- LBM\_MSG\_FLUSH - Message is to be sent ASAP (not implicitly or explicitly batched). This also flushes waiting messages that were explicitly or implicitly batched.
- LBM\_SRC\_NONBLOCK - If message could not be sent immediately return and error and signal LBM\_EWOULDBLOCK.
- LBM\_SRC\_BLOCK - Block the caller indefinitely until the message is sent. (This behavior is the default if neither LBM\_SRC\_NONBLOCK nor LBM\_SRC\_BLOCK are supplied.)

*exinfo* Pointer to `lbm_src_send_ex_info_t` options that includes the 32 or 64 bit hot-failover sequence number to send.

**Returns:**

-1 for Failure or 0 for Success.

#### 8.1.4.121 LBMEExpDLL int `lbt_src_send_recv_reset` (`lbt_src_t * src, int flags, lbt_src_send_ex_info_t * exinfo`)

Send a message that instructs hot-failover receivers to reset their state. In, and only in, the case that hf receivers cannot be manually restarted, this function can be used to allow delivering of previously sent sequence numbers. The hot-failover receiver will deliver a message of type `LBM_MSG_HF_RESET` and will include the new expected sequence number. The sequence number contained with the reset will be used as the next expected sequence number to be sent. The LBM source must have been created with `lbt_src_create` and not `lbt_src_create`.

NOTE: The best way to reset a hot-failover receiver's state is to restart the receiver itself. This function should be used only when that is impossible.

**Parameters:**

*src* Pointer to the LBM source to send from, must be a hot failover source  
*exinfo* Pointer to the `lbt_src_send_ex_info_t` containing the hf sequence number

**Returns:**

-1 for Failure or 0 for Success

**8.1.4.122 LBMExpDLL int lbt\_hf\_src\_sendv (lbt\_src\_t \* src, const lbt\_iovec\_t \* iov, int num, lbt\_uint\_t sqn, int flags)**

The LBM source must have been created with `lbt_hf_src_create` and not `lbt_src_create`. The message is specified as an array of iovecs.

NOTE: Unlike `lbt_src_sendv`, which by default sends N number of messages where N is the length of the iovec; `lbt_hf_src_sendv` will gather the elements of the array into one message.

**See also:**

[lbt\\_hf\\_src\\_sendv](#)

**Warning:**

It is not recommended to call this function from a context thread callback. If called from a context thread callback, use the `LBM_SRC_NONBLOCK` flag and handle any `LBM_EWOULDBLOCK` errors internally.

**Parameters:**

*src* Pointer to the LBM source to send from.  
*iov* Pointer to an array of iovecs that hold message information.  
*num* Number of elements of the iov array to send.  
*sqn* The application sequence number to associate with this message.  
*flags* Flags indicating various conditions. ORed set of values.

- `LBM_MSG_START_BATCH` - Messages start a batch of messages
- `LBM_MSG_END_BATCH` - Messages end a batch of messages. Batch should be sent to the implicit batching buffer.
- `LBM_MSG_COMPLETE_BATCH` - Messages constitute a complete batch and should be sent to the implicit batching buffer.
- `LBM_MSG_FLUSH` - Messages are to be sent ASAP (not implicitly batched or explicitly batched).
- `LBM_SRC_NONBLOCK` - If messages could not be sent immediately return and error and signal `LBM_EWOULDBLOCK`.

- LBM\_SRC\_BLOCK - Block the caller indefinitely until the messages are all sent. (This behavior is the default if neither LBM\_SRC\_NONBLOCK nor LBM\_SRC\_BLOCK are supplied.)

**Returns:**

-1 for Failure or 0 for Success.

#### **8.1.4.123 LBMEExpDLL int lbm\_hf\_src\_sendv\_ex (lbm\_src\_t \* src, const lbm\_ovec\_t \* iov, int num, lbm\_uint\_t sqn, int flags, lbm\_src\_send\_ex\_info\_t \* exinfo)**

The LBM source must have been created with lbm\_hf\_src\_create and not lbm\_src\_create. The message is specified as an array of iovecs.

NOTE: Unlike lbm\_src\_sendv, which by default sends N number of LBM Messages where N is the length of the iovec array; lbm\_hf\_src\_sendv will gather the elements of the array into a single message.

**See also:**

[lbt\\_hf\\_src\\_sendv\\_ex](#)

**Warning:**

If called from a context thread callback, use the LBM\_SRC\_NONBLOCK flag and handle any LBM\_EWOULDLOCK errors internally.

Calling this function from a context thread callback for stability and confirmation events could cause a deadlock

**Parameters:**

*src* Pointer to the LBM source to send from

*iov* Pointer to an array of iovecs that hold message information.

*num* Number of elements of the iov array to send.

*sqn* The application sequence number to associate with this message.

*flags* Flags indicating various conditions. ORed set of values.

- LBM\_MSG\_START\_BATCH - Message starts a batch of messages
- LBM\_MSG\_END\_BATCH - Message ends a batch of messages. Batch should be sent to the implicit batching buffer.
- LBM\_MSG\_COMPLETE\_BATCH - Message constitutes a complete batch and should be sent to the implicit batching buffer.
- LBM\_MSG\_FLUSH - Message is to be sent ASAP (not implicitly or explicitly batched). This also flushes waiting messages that were explicitly or implicitly batched.

- LBM\_SRC\_NONBLOCK - If message could not be sent immediately return and error and signal LBM\_EWOULDBLOCK.
- LBM\_SRC\_BLOCK - Block the caller indefinitely until the message is sent. (This behavior is the default if neither LBM\_SRC\_NONBLOCK nor LBM\_SRC\_BLOCK are supplied.)

***exinfo*** Pointer to `lbt_src_send_ex_info_t` options which includes the 32 or 64 bit hot-failover sequence number to send.

**Returns:**

-1 for Failure or 0 for Success.

#### **8.1.4.124 LBMEExpDLL int `lbt_hfx_attr_create` (`lbt_hfx_attr_t ** attr`)**

The attribute object is allocated and filled with the current default values that are used by `lbt_hfx_t` objects and may have been modified by a previously loaded configuration file.

**Parameters:**

***attr*** A pointer to a pointer to a UM hfx attributes structure. Will be filled in by this function to point to the newly created `lbt_hfx_attr_t` object.

**Returns:**

0 for Success and -1 for Failure

#### **8.1.4.125 LBMEExpDLL int `lbt_hfx_attr_create_default` (`lbt_hfx_attr_t ** attr`)**

The attribute object is allocated and filled with the initial or factory default values built into LBM that are used by `lbt_hfx_t` objects.

**Parameters:**

***attr*** A pointer to a pointer to a UM hfx attribute structure. Will be filled in by this function to point to the newly created `lbt_hfx_attr_t` object.

**Returns:**

0 for Success and -1 for Failure

**8.1.4.126 LBMExpDLL int *lbfm\_hfx\_attr\_create\_from\_xml* (*lbfm\_hfx\_attr\_t* \*\**attr*, const char \* *topicname*)**

The attribute object is allocated and filled with the current default values that are used by *lbfm\_hfx\_t* objects and may have been modified by a previously loaded configuration file. Then, if an XML configuration file has been loaded, the attribute object is further filled with the defaults for the given topic name. If the topic name is not permitted by the XML configuration, -1 is returned and no attribute object is created.

**Parameters:**

*attr* A pointer to a pointer to a UM hfx attribute structure. Will be filled in by this function to point to the newly created *lbfm\_hfx\_attr\_t* object.

*topicname* The topic name used to lookup this topic in the XML configuration.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.127 LBMExpDLL int *lbfm\_hfx\_attr\_delete* (*lbfm\_hfx\_attr\_t* \* *attr*)**

The attribute object is cleaned up and deleted.

**Parameters:**

*attr* Pointer to a UM hfx attribute object as returned by [lbfm\\_hfx\\_attr\\_create](#).

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.128 LBMExpDLL int *lbfm\_hfx\_attr\_dump* (*lbfm\_hfx\_attr\_t* \* *attr*, int \* *size*, *lbfm\_config\_option\_t* \* *opts*)**

The config object is filled with HFX configuration options

**Parameters:**

*cattr* The HFX attribute object to retrieve the attributes from

*size* Size of the *opts* array. Will return the number of items that were set in *opts*

*opts* The options array to fill

**8.1.4.129 LBMExpDLL int lhm\_hfx\_attr\_dup (lhm\_hfx\_attr\_t \*\*attr, const lhm\_hfx\_attr\_t \*original)**

A new attribute object is created as a copy of an existing object.

**Parameters:**

*attr* A pointer to a pointer to a UM hfx attribute structure. Will be filled in by this function to point to the newly created lhm\_hfx\_attr\_t object.

*original* Pointer to a UM hfx attribute object as returned by [lhm\\_hfx\\_attr\\_create](#) or [lhm\\_hfx\\_attr\\_create\\_default](#), from which *attr* is initialized.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.130 LBMExpDLL int lhm\_hfx\_attr\_getopt (lhm\_hfx\_attr\_t \*attr, const char \*optname, void \*optval, size\_t \*optlen)****Parameters:**

*attr* Pointer to a UM hfx attributed object.

*optname* String containing the option name.

*optval* Pointer to the option value structure to be filled. The structure of the option values are specific to the options themselves.

*optlen* Length (in bytes) of the *optval* structure when passed in. Upon return, this is set to the size of the optval filled in structure.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.131 LBMExpDLL int lhm\_hfx\_attr\_option\_size ()****Returns:**

The number of entries that are of type "hfx"

**8.1.4.132 LBMExpDLL int lhm\_hfx\_attr\_set\_from\_xml (lhm\_hfx\_attr\_t \*attr, const char \*topicname)**

The attribute object is filled with the default values for the given topic name, if an XML configuration file has been loaded. If the topic name is not permitted by the XML configuration, -1 is returned and no values are set.

**Parameters:**

*attr* A pointer to a UM hfx attribute structure.

*topicname* The topic name used to lookup this topic in the XML configuration.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.133 LBMEExpDLL int lbm\_hfx\_attr\_setopt (lbm\_hfx\_attr\_t \* attr, const char \* optname, const void \* optval, size\_t optlen)**

Used before the hfx is created. NOTE: the attribute object must first be initialized with the corresponding \_attr\_create() function.

**Parameters:**

*attr* Pointer to a UM hfx attribute object.

*optname* String containing the option name.

*optval* Pointer to the option value structure. The structure of the option values are specific to the options themselves.

*optlen* Length (in bytes) of the *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.134 LBMEExpDLL int lbm\_hfx\_attr\_str\_getopt (lbm\_hfx\_attr\_t \* attr, const char \* optname, char \* optval, size\_t \* optlen)****Parameters:**

*attr* Pointer to a UM hfx attributed object.

*optname* String containing the option name.

*optval* Pointer to the string to be filled in.

*optlen* Maximum length (in bytes) of the *string* when passed in. Upon return, this is set to the size of the formatted string.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.135 LBMExpDLL int lbm\_hfx\_attr\_str\_setopt (lbm\_hfx\_attr\_t \* attr,  
const char \* optname, const char \* optval)**

Used before the hfx is created. NOTE: the attribute object must first be initialized with the corresponding \_attr\_create() function.

**Parameters:**

*attr* Pointer to a UM hfx attributed object.

*optname* String containing the option name.

*optval* String containing the option value. The format of the string is specific to the option itself.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.136 LBMExpDLL int lbm\_hfx\_create (lbm\_hfx\_t \*\* hfxp,  
lbm\_hfx\_attr\_t \* cattr, const char \* symbol, lbm\_rev\_cb\_proc proc,  
lbm\_event\_queue\_t \* evq)****See also:**

[lbm\\_hfx\\_delete\(\)](#)

**Parameters:**

*hfxp* A pointer to a pointer to a UM hfx object. Will be filled in by this function to point to the newly created lbm\_hfx\_t object.

*cattr* A pointer to a UM hfx attribute object. A value of NULL will use default attributes.

*symbol* The symbol string to be used for all hot failover receivers managed by this HFX.

*proc* Pointer to a function to call when messages arrive.

*evq* Optional Event Queue to place message events on when they arrive. If NULL, causes *proc* to be called from context thread.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.137 LBMEExpDLL int lbm\_hfx\_delete (lbm\_hfx\_t \* *hfx*)****Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*hfx* Pointer to a UM hfx object to delete.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.138 LBMEExpDLL int lbm\_hfx\_delete\_ex (lbm\_hfx\_t \* *hfx*,  
lbm\_event\_queue\_cancel\_cb\_info\_t \* *cbinfo*)****See also:**

[lbm\\_hf\\_rcv\\_delete\\_ex](#) or

[lbm\\_rcv\\_delete\\_ex](#), this extended callback can be used whether or not an event queue is associated with the HFX.

**Warning:**

It is not safe to call this function from a context thread callback.

When deleting an hfx object, wait for the delete\_ex callback before deleting any of the associated contexts.

**Parameters:**

*hfx* Pointer to an LBM hfx object to delete.

*cbinfo* Cancellation callback information containing the (optional) event queue, function pointer, and client data for the callback.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.139 LBMEExpDLL int lbm\_hfx\_dump (lbm\_hfx\_t \* *hfx*, int \* *size*,  
lbm\_config\_option\_t \* *opts*)**

The config object is filled with HFX configuration options

**Parameters:**

*hfx* The HFX object to retrieve the attributes from

*size* Size of the opts array. Will return the number of items that were set in opts

*opts* The options array to fill

#### 8.1.4.140 LBMEExpDLL int **lbt\_hfx\_getopt** (*lbt\_hfx\_t* \* *hfx*, const char \* *optname*, void \* *optval*, size\_t \* *optlen*)

**Parameters:**

*hfx* Pointer to a UM hfx object where the option is stored.

*optname* String containing the option name.

*optval* Pointer to the option value structure. The structure of the option values are specific to the options themselves.

*optlen* When passed, this is the max length (in bytes) of the *optval* structure.  
When returned, this is the length of the filled in *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

#### 8.1.4.141 LBMEExpDLL int **lbt\_hfx\_recv\_create** (*lbt\_hfx\_recv\_t* \*\* *hfrcvp*, *lbt\_hfx\_t* \* *hfx*, *lbt\_context\_t* \* *ctx*, *lbt\_recv\_topic\_attr\_t* \* *rattr*, void \* *clientd*)

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*hfrcvp* A pointer to a pointer to a UM hfx\_recv\_t object. Will be filled in by this function to point to the newly created lbt\_hfx\_recv\_t object.

*hfx* An lbt\_hfx\_t object created by

**See also:**

[lbt\\_hfx\\_create](#)

**Parameters:**

*ctx* The lbt\_context\_t object on which to create the new receiver.

*rattr* The receiver attributes to be used when creating new hot failover receivers.

*clientd* Pointer to client data to be delivered when a message is received and the lbt\_hfx\_t object's proc is called.

**8.1.4.142 LBMExpDLL int lbm\_hfx\_rcv\_delete (lbm\_hfx\_rcv\_t \* *hfrcv*)****Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*hfrcv* Pointer to a UM HFX receiver object to delete.

**Returns:**

0 for Success and -1 for Failure

**8.1.4.143 LBMExpDLL int lbm\_hfx\_rcv\_delete\_ex (lbm\_hfx\_rcv\_t \* *hfrcv*,  
lbm\_event\_queue\_cancel\_cb\_info\_t \* *cinfo*)**

Delete a UM Hot Failover receiver object, with an application callback indicating when the receiver is fully cancelled. This extended version of the receiver delete function requires the configuration option queue\_cancellation\_callbacks\_enabled to be set to 1 if an event queue is in use.

Unlike

**See also:**

[lbm\\_hf\\_rcv\\_delete\\_ex](#) or

[lbm\\_rcv\\_delete\\_ex](#), this extended callback can be used whether or not an event queue is associated with the HFX. This allows an application to delete a receiver from a single context and be notified when any messages currently held in the order map are no longer required.

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*hfrcv* Pointer to a UM HFX receiver to delete.

*cinfo* Cancellation callback information containing the (optional) event queue, function pointer, and client data for the callback.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.144 LBMEExpDLL int lhm\_hfx\_rev\_topic\_dump (lhm\_hfx\_rev\_t \* *hfxrcv*,  
int \* *size*, lhm\_config\_option\_t \* *opts*)**

The config object is filled with receiver configuration options

**Parameters:**

*hfxrcv* The HFX receiver object to retrieve the attributes from

*size* Size of the opts array. Will return the number of items that were set in opts

*opts* The options array to fill

**8.1.4.145 LBMEExpDLL int lhm\_hfx\_setopt (lhm\_hfx\_t \* *hfx*, const char \*  
*optname*, const void \* *optval*, size\_t *optlen*)**

Only those options that can be set during operation may be specified. See "Options That May Be Set During Operation" (doc/Config/maybesetduringoperation.html) in The UM Configuration Guide. For API functions that can access any option, see lhm\_hfx\_attr\_().

**Parameters:**

*hfx* Pointer to a UM hfx object where the option is to be set.

*optname* String containing the option name.

*optval* Pointer to the option value structure. The structure of the option values are specific to the options themselves.

*optlen* Length (in bytes) of the *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.146 LBMEExpDLL int lhm\_hfx\_str\_getopt (lhm\_hfx\_t \* *hfx*, const char \*  
*optname*, char \* *optval*, size\_t \* *optlen*)**

**Parameters:**

*hfx* Pointer to a UM hfx object where the option is stored.

*optname* String containing the option name.

*optval* String to hold the option value.

*optlen* When passed, this is the max length (in bytes) of the string. When returned, this is the length of the filled in option value.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.147 LBMEExpDLL int lbm\_hfx\_str\_setopt (lbm\_hfx\_t \* *hfx*, const char \* *optname*, const char \* *optval*)**

Only those options that can be set during operation may be specified. See "Options That May Be Set During Operation" (doc/Config/maybesetduringoperation.html) in The UM Configuration Guide. For API functions that can access any option, see *lbm\_hfx\_attr\_\**().

**Parameters:**

*hfx* Pointer to a UM hfx object where the option is to be set.

*optname* String containing the option name.

*optval* String containing the option value. The format of the string is specific to the options themselves.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.148 LBMEExpDLL int lbm\_is\_ume\_capable (void)****Returns:**

1 if the library is capable of UME operations, 0 if the library is not capable of UME operations.

**8.1.4.149 LBMEExpDLL int lbm\_is\_umq\_capable (void)****Returns:**

1 if the library is capable of UMQ operations, 0 if the library is not capable of UMQ operations.

**8.1.4.150 LBMEExpDLL int lbm\_license\_file (const char \* *licfile*)****Parameters:**

*licfile* String containing the name of a file that contains the UM license. This string is the same as that which would otherwise be specified as the value of the LBM\_LICENSE\_FILENAME environmental variable.

**Returns:**

0 for Success and -1 for Failure

**8.1.4.151 LBMExpDLL int lbm\_license\_str (const char \* *licstr*)****Parameters:**

*licstr* String containing the UM license. This string is the same as that which would otherwise be specified as the value of the LBM\_LICENSE\_INFO environmental variable.

**Returns:**

0 for Success and -1 for Failure

**8.1.4.152 LBMExpDLL int lbm\_log (lbm\_log\_cb\_proc *proc*, void \* *clientd*)****Parameters:**

*proc* Function to call when a log message is generated.

*clientd* Client data to pass when a log message is generated.

**Returns:**

0 on Success and -1 for Failure

**8.1.4.153 LBMExpDLL void lbm\_logf (int *level*, const char \* *format*, ...)****Parameters:**

*level* Message log level (see LBM\_LOG\_\*).

*format* printf style format string, followed by zero or more arguments.

**8.1.4.154 LBMExpDLL int lbm\_msg\_delete (lbm\_msg\_t \* *msg*)**

This should only be called if the message was previously saved via [lbm\\_msg\\_retain\(\)](#). Any associated lbm\_response\_t objects for this message are cleaned up automatically in this function.

**Note:**

A receive callback should never delete the message that was passed in. It should either let UM delete it when the callback returns, or it should retain it and delete it later.

**Parameters:**

*msg* Pointer to a UM message object to delete.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.155 LBMEExpDLL int lbm\_ume\_rcv\_ack\_t\* lbm\_msg\_extract\_ume\_ack (lbm\_msg\_t \* msg)****See also:**

[lbm\\_ume\\_ack\\_delete](#)

**Parameters:**

*msg* Pointer to the message object from which to extract the ack structure.

**Returns:**

the ack structure for Success, NULL for failure.

**8.1.4.156 LBMEExpDLL int lbm\_msg\_is\_fragment (lbm\_msg\_t \* msg)****Parameters:**

*msg* Pointer to a UM message object to retrieve fragment information from.

*info* Pointer to fragment information structure to fill in.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.157 LBMEExpDLL int lbm\_msg\_properties\_clear (lbm\_msg\_properties\_t \* properties, const char \* name)****See also:**

[lbm\\_msg\\_properties\\_create](#)

[lbm\\_msg\\_properties\\_set](#)

[lbm\\_msg\\_properties\\_get](#)

**Parameters:**

*properties* Properties object from which the named property should be cleared.

*name* Property to be cleared.

**Returns:**

LBM\_OK for success, LBM\_FAILURE if the property was not present.

**8.1.4.158 LBMEExpDLL int lbm\_msg\_properties\_create (lbm\_msg\_properties\_t \*  
\*\**properties*)**

**See also:**

[lbm\\_src\\_send\\_ex](#)  
[lbm\\_msg\\_properties\\_delete](#)  
[lbm\\_msg\\_properties\\_set](#)  
[lbm\\_msg\\_properties\\_get](#)

**Parameters:**

*properties* A pointer to a pointer to be filled in by this function.

**Returns:**

LBM\_OK for success, LBM\_FAILURE for failure if the memory cannot be allocated.

**8.1.4.159 LBMEExpDLL int lbm\_msg\_properties\_delete (lbm\_msg\_properties\_t \*  
\**properties*)**

**See also:**

[lbm\\_src\\_send\\_ex](#)  
[lbm\\_msg\\_properties\\_create](#)  
[lbm\\_msg\\_properties\\_set](#)  
[lbm\\_msg\\_properties\\_get](#)

**Parameters:**

*properties* A pointer to a properties object.

**Returns:**

LBM\_OK for success, LBM\_FAILURE for failure.

**8.1.4.160 LBMEExpDLL int lbm\_msg\_properties\_get (lbm\_msg\_properties\_t \*  
\**properties*, const char \**name*, void \**value*, int \**type*, size\_t \**size*)**

**See also:**

[lbm\\_src\\_send\\_ex](#)  
[lbm\\_msg\\_properties\\_create](#)  
[lbm\\_msg\\_properties\\_get](#)

**Parameters:**

*properties* The properties object that the new value should be retrieved from

*name* The name of the property to be retrieved

*value* A pointer to the memory to be filled in with the value.

*type* A pointer to the type the value should be retrieved as. If the specified type is not compatible with the property, this field will be filled in with the required type.

*size* A pointer to a size\_t holding the size of the memory block available to be filled in. If a block of insufficient size is specified, this field will be filled in with the required size. For string types, the block of memory must be of sufficient size to hold the string as well as the null terminator.

**Returns:**

LBM\_OK for success, LBM\_FAILURE for failure, and changes the current value of [lbtm\\_errnum\(\)](#) and [lbtm\\_errstr\(\)](#).

**8.1.4.161 LBMEExpDLL int lbtm\_msg\_properties\_iter\_create  
(lbtm\_msg\_properties\_iter\_t \*\*iterp)****See also:**

[lbtm\\_msg\\_properties\\_iter\\_first](#) to begin iterating over a properties object.

**Parameters:**

*iterp* A pointer to a pointer that will be filled in with the newly-created iterator object.

**Returns:**

LBM\_OK for success, LBM\_FAILURE for failure.

**8.1.4.162 LBMEExpDLL int lbtm\_msg\_properties\_iter\_delete  
(lbtm\_msg\_properties\_iter\_t \*iter)****See also:**

[lbtm\\_msg\\_properties\\_iter\\_create](#)

**Parameters:**

*iter* A pointer to an iterator created via

See also:

[lbm\\_msg\\_properties\\_iter\\_create](#)

Returns:

LBM\_OK for success, LBM\_FAILURE for failure.

#### 8.1.4.163 LBMEExpDLL int lbm\_msg\_properties\_iter\_first (*lbm\_msg\_properties\_iter\_t \*iter, lbm\_msg\_properties\_t \*properties*)

See also:

*lbm\_msg\_properties\_t* object, starting at the first element. Calling [lbpmsg\\_properties\\_iter\\_first](#) associates an iterator with a properties object, and sets its current position to the first property available. An iterator can be used to iterate over more than one properties object as long as [lbpmsg\\_properties\\_iter\\_first](#) is called to associate it with each new properties object.

[lbpmsg\\_properties\\_iter\\_next](#)

Parameters:

*iter* An iterator object allocated via

See also:

[lbpmsg\\_properties\\_iter\\_create](#)

Parameters:

*properties* A properties object, either retrieved from an *lbm\_msg\_t*, or created via

See also:

[lbpmsg\\_properties\\_create](#)

Returns:

LBM\_OK for success, LBM\_FAILURE if there are no elements contained in the properties object.

#### 8.1.4.164 LBMEExpDLL int lbm\_msg\_properties\_iter\_next (*lbm\_msg\_properties\_iter\_t \*iter*)

See also:

*lbm\_msg\_properties\_t* object.  
[lbpmsg\\_properties\\_iter\\_first](#)  
[lbpmsg\\_properties\\_iter\\_prev](#)

**Parameters:**

*iter* Iterate to the next element in the currently associated `lbt_msg_properties_t` object.

**Returns:**

`LBM_OK` for success, `LBM_FAILURE` if the iterator already points to the last element.

**8.1.4.165 LBMEExpDLL int lbt\_msg\_properties\_set (lbt\_msg\_properties\_t \* properties, const char \* name, const void \* value, int type, size\_t size)****See also:**

[lbt\\_src\\_send\\_ex](#)  
[lbt\\_msg\\_properties\\_create](#)  
[lbt\\_msg\\_properties\\_get](#)

**Parameters:**

*properties* The properties object that the new value should be set on

*name* The name of the property to be set

*value* The value to set.

*type* The type of value being specified.

*size* The size of the value being specified. For string types, the specified number of bytes will be copied, and a null terminator will be appended.

**Returns:**

`LBM_OK` for success, `LBM_FAILURE` for failure, and changes the current value of [lbt\\_errnum\(\)](#) and [lbt\\_errstr\(\)](#).

**8.1.4.166 LBMEExpDLL int lbt\_msg\_retain (lbt\_msg\_t \* msg)**

This function should be called from inside a receiver callback function to prevent UM from automatically deleting the message when the callback function returns (LBM's normal behavior).

Once retained, the application has the responsibility to dispose of the message when it is finished with it by calling [lbt\\_msg\\_delete\(\)](#).

**Parameters:**

*msg* Pointer to a UM message object to retain.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.167 LBMEExpDLL int lbm\_msg\_retrieve\_fragment\_info (lbm\_msg\_t \*  
*msg*, lbm\_msg\_fragment\_info\_t \**info*)****Parameters:**

*msg* Pointer to a UM message object

**Returns:**

1 for Success and 0 for Failure.

**8.1.4.168 LBMEExpDLL int lbm\_msg\_retrieve\_gateway\_info (lbm\_msg\_t \**msg*,  
lbm\_msg\_gateway\_info\_t \**info*)****Parameters:**

*msg* Pointer to a UM message object to retrieve gateway information from.

*info* Pointer to gateway information structure to fill in.

**Returns:**

0 for Success and -1 for Failure.

**Deprecated****8.1.4.169 LBMEExpDLL int lbm\_msg\_retrievemsgid (lbm\_msg\_t \**msg*,  
lbm\_umq\_msgid\_t \**id*)****Parameters:**

*msg* Pointer to a UM message object to retrieve UMQ Message ID info from.

*id* Pointer to UMQ Message ID structure to fill in.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.170 LBMEExpDLL int lbm\_msg\_retrieve\_umq\_index (lbm\_msg\_t \* *msg*,  
                  lbm\_umq\_index\_info\_t \* *info*)**

**Parameters:**

*msg* Pointer to a UM message object to retrieve UMQ index info from.  
*info* Pointer to UMQ index structure to fill in.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.171 LBMEExpDLL int lbm\_msg\_ume\_can\_send\_explicit\_ack (lbm\_msg\_t  
                  \* *msg*)**

**Parameters:**

*msg* Pointer to a UM message object to acknowledge up to.

**Returns:**

1 for true and 0 for False.

**8.1.4.172 LBMEExpDLL int lbm\_msg\_ume\_send\_explicit\_ack (lbm\_msg\_t \*  
                  *msg*)**

This function causes a UMP Explicit ACK to be sent that acknowledges previous messages since the last UMP Explicit ACK for the source was performed.

**Parameters:**

*msg* Pointer to a UM message object to acknowledge up to.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.173 LBMEExpDLL int lbm\_msg\_umq\_reassign (lbm\_msg\_t \* *msg*, int  
                  *flags*)**

**Parameters:**

*msg* Pointer to a UM message to request to be reassigned.

*flags* Flags indicating various conditions. ORed set of values.

- LBM\_MSG\_UMQ\_REASSIGN\_FLAG\_DISCARD - Message should be discarded instead of being assigned.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.174 LBMEExpDLL int lbm\_multicast\_immediate\_message (lbm\_context\_t \* ctx, const char \* topic, const char \* data, size\_t len, int flags)****Warning:**

Multicast immediate messages are NOT guaranteed to maintain order. A loss-recovery event can lead to messages received out of order.

**Parameters:**

*ctx* Pointer to UM context to send from

*topic* Topic name to send message to or NULL for non-topic. Topic names should be limited to 246 characters (not including the final null).

*data* Pointer to the data to send in this message

*len* Length (in bytes) of the data to send in this message. Multicast immediate messages must be 7866 bytes or less in length.

*flags* Flags indicating various conditions. ORed set of values.

- LBM\_SRC\_NONBLOCK - If message could not be sent immediately return and error and signal LBM\_EWOULDBLOCK.
- LBM\_SRC\_BLOCK - Block the caller indefinitely until the message is sent. (This behavior is the default if neither LBM\_SRC\_NONBLOCK nor LBM\_SRC\_BLOCK are supplied.)
- LBM\_MSG\_FLUSH - Messages are to be sent ASAP (not implicitly batched).

**Returns:**

-1 for Failure or 0 for Success.

**8.1.4.175 LBMEExpDLL int lbm\_multicast\_immediate\_request (lbm\_request\_t \*\* reqp, lbm\_context\_t \* ctx, const char \* topic, const char \* data, size\_t len, lbm\_request\_cb\_proc proc, void \* clientd, lbm\_event\_queue\_t \* evq, int flags)****Warning:**

Multicast immediate messages are NOT guaranteed to maintain order. A loss-recovery event can lead to messages received out of order.

**Parameters:**

***reqp*** A pointer to a pointer for the `lbt_request_t` object created to be stored.

***ctx*** Pointer to UM context to send from.

***topic*** Topic name to send message to or NULL for non-topic. Topic names should be limited to 246 characters (not including the final null).

***data*** Buffer to be included as data in the request.

***len*** Length (in bytes) of the data to send in this message. Multicast immediate messages must be 7866 bytes or less in length.

***proc*** Pointer to function to call when responses come in for this request.

***clientd*** Client data returned in the callback proc *proc*.

***evq*** Optional Event Queue to place message events on when they occur. If NULL causes *proc* to be called from context thread.

***flags*** Flags used to instruct UM how to handle this message. See *lbt\_multicast\_immediate\_message* for more information.

**Returns:**

0 for Success and -1 for Failure.

---

**8.1.4.176 LBMExpDLL int lbt\_queue\_immediate\_message (lbt\_context\_t \*  
                  ctx, const char \* qname, const char \* topic, const char \* data, size\_t  
                  len, int flags, lbt\_src\_send\_ex\_info\_t \* info)**
**Parameters:**

***ctx*** Pointer to UM context to submit from.

***qname*** Queue to submit message to. Queue names should be limited to 246 bytes characters (not including the final NULL).

***topic*** Topic name to send message to. Topic names should be limited to 246 characters (not including the final NULL).

***data*** Pointer to the data to send in this message

***len*** Length (in bytes) of the data to send in this message.

***flags*** Flags used to instruct UM how to handle this message. See *lbt\_unicast\_immediate\_message* for more information.

***info*** Pointer to `lbt_src_send_ex_info_t` options

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.177 LBMEExpDLL int lbt\_rcv\_create (lbt\_rcv\_t \*\**rcvp*, lbt\_context\_t \**ctx*, lbt\_topic\_t \**topic*, lbt\_rcv\_cb\_proc *proc*, void \**clientd*, lbt\_event\_queue\_t \**evq*)**

The callback *proc* will be called to deliver data sent to the topics that the receiver has requested.

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*rcvp* A pointer to a pointer to a UM receiver object. Will be filled in by this function to point to the newly created lbt\_rcv\_t object.

*ctx* Pointer to the UM context object associated with the receiver.

*topic* Pointer to the UM topic object associated with the desired receiver topic.

**Warning:**

Topic references should not be reused. Each [lbt\\_rcv\\_create\(\)](#) call should be preceded by a call to [lbt\\_rcv\\_topic\\_lookup\(\)](#).

**Parameters:**

*proc* Pointer to a function to call when messages arrive.

*clientd* Pointer to client data that is passed when data arrives and *proc* is called.

*evq* Optional Event Queue to place message events on when they arrive. If NULL causes *proc* to be called from context thread.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.178 LBMEExpDLL int lbt\_rcv\_delete (lbt\_rcv\_t \**rcv*)**

Delete a UM receiver object. Note that there are rare circumstances where this function can return while the receiver callback may still be executing. This would only occur if receiver events are being delivered via an event queue. If the application needs to know when all possible processing on the receiver is complete, it must use [lbt\\_rcv\\_delete\\_ex\(\)](#).

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*rcv* Pointer to a UM receiver object to delete.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.179 LBMEExpDLL int lbm\_rcv\_delete\_ex (lbm\_rcv\_t \* *rcv*,  
                          lbm\_event\_queue\_cancel\_cb\_info\_t \* *cinfo*)**

Delete a UM receiver object, with an application callback indicating when the receiver is fully canceled. This extended version of the receiver delete function requires the configuration option queue\_cancellation\_callbacks\_enabled be set to 1.

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*rcv* Pointer to a UM receiver object to delete.

*cinfo* Cancellation callback information, containing the event queue, function pointer, and client data for the callback.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.180 LBMEExpDLL lbm\_rcv\_t\* lbm\_rcv\_from\_hf\_recv (lbm\_hf\_recv\_t \*  
                          *hfrecv*)****Parameters:**

*hfrecv* Pointer to a UM HF receiver object.

**Returns:**

Pointer to a UM receiver for the LBM HF receiver object.

**8.1.4.181 LBMEExpDLL lbm\_rcv\_t\* lbm\_rcv\_from\_hfx\_recv (lbm\_hfx\_recv\_t \*  
                          *hfxrecv*)****Parameters:**

*hfxrecv* A pointer to a UM hfx\_recv\_t object.

**8.1.4.182 LBMEExpDLL int lbm\_rcv\_getopt (lbm\_rev\_t \* *rcv*, const char \* *optname*, void \* *optval*, size\_t \* *optlen*)**

**Parameters:**

*rcv* Pointer to a UM receiver object where the option is stored

*optname* String containing the option name

*optval* Pointer to the option value structure. The structure of the option values are specific to the options themselves.

*optlen* When passed, this is the max length (in bytes) of the *optval* structure.  
When returned, this is the length of the filled in *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.183 LBMEExpDLL int lbm\_rcv\_msg\_source\_clientd (lbm\_rev\_t \* *rcv*, const char \* *source*, void \* *source\_clientd*)**

**Parameters:**

*rcv* Pointer to the UM receiver to look for the source on.

*source* String version of the source to look for.

*source\_clientd* Pointer value to set in subsequent messages delivered.

**Returns:**

-1 for Failure and 0 for Success.

**8.1.4.184 LBMEExpDLL int lbm\_rcv\_reset\_all\_transport\_stats (lbm\_rev\_t \* *rcv*)**

**Parameters:**

*rcv* Pointer to the UM receiver to reset statistics for.

**Returns:**

-1 for Failure and 0 for Success.

**8.1.4.185 LBMEExpDLL int lbm\_rcv\_reset\_transport\_stats (lbm\_rcv\_t \* *rcv*, const char \* *source*)****Parameters:**

*rcv* Pointer to the UM receiver to reset statistics for.

*source* String version of the source to reset statistics for.

**Returns:**

-1 for Failure and 0 for Success.

**8.1.4.186 LBMEExpDLL int lbm\_rcv\_retrieve\_all\_transport\_stats (lbm\_rcv\_t \* *rcv*, int \* *num*, lbm\_rcv\_transport\_stats\_t \* *stats*)****Parameters:**

*rcv* Pointer to the UM receiver to retrieve statistics for.

*num* Pointer to an integer that must hold the maximum number of elements in the stats array when passed in. Upon return, this value is set to the number of sources filled in.

*stats* Array of lbm\_rcv\_transport\_stats\_t objects to fill in transport stats for.

**Returns:**

-1 for Failure and 0 for Success.

**Note:**

If -1 is returned, and [lbm\\_ernum\(\)](#) returns LBM\_EINVAL, then *\*num* may contain a larger number than the value originally passed into this function. This return value represents the number of lbm\_rcv\_transport\_stats\_t objects required in the *stats* array and can be used to dynamically determine how many entries are needed.

**8.1.4.187 LBMEExpDLL int lbm\_rcv\_retrieve\_transport\_stats (lbm\_rcv\_t \* *rcv*, const char \* *source*, lbm\_rcv\_transport\_stats\_t \* *stats*)****Parameters:**

*rcv* Pointer to the UM receiver to retrieve statistics for.

*source* String version of the source to retrieve stats for.

*stats* Pointer to a stats structure to fill in.

**Returns:**

-1 for Failure and 0 for Success.

**8.1.4.188 LBMEExpDLL int lbm\_rcv\_setopt (lbm\_rcv\_t \* *rcv*, const char \* *optname*, const void \* *optval*, size\_t *optlen*)**

Only those options that can be set during operation may be specified. See "Options That May Be Set During Operation" (doc/Config/maybesetduringoperation.html) in The UM Configuration Guide. For API functions that can access any option, see *lbm\_rcv\_topic\_attr\_\*()*.

**Parameters:**

- rcv*** Pointer to a UM receiver object where the option is to be set
- optname*** String containing the option name
- optval*** Pointer to the option value structure. The structure of the option values are specific to the options themselves.
- optlen*** Length (in bytes) of the *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.189 LBMEExpDLL int lbm\_rcv\_str\_getopt (lbm\_rcv\_t \* *rcv*, const char \* *optname*, char \* *optval*, size\_t \* *optlen*)****Parameters:**

- rcv*** Pointer to a UM receiver object where the option is stored
- optname*** String containing the option name
- optval*** String to be filled in with the option value.
- optlen*** When passed, this is the max length (in bytes) of the string. When returned, this is the length of the filled in with the option value.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.190 LBMEExpDLL int lbm\_rcv\_str\_setopt (lbm\_rcv\_t \* *rcv*, const char \* *optname*, const char \* *optval*)**

Only those options that can be set during operation may be specified. See "Options That May Be Set During Operation" (doc/Config/maybesetduringoperation.html) in The UM Configuration Guide. For API functions that can access any option, see *lbm\_rcv\_topic\_attr\_\*()*.

**Parameters:**

*rcv* Pointer to a UM receiver object where the option is to be set  
*optname* String containing the option name  
*optval* String containing the option value. The format of the string is specific to the options themselves.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.191 LBMEExpDLL int lbm\_rcv\_subscribe\_channel (lbm\_rcv\_t \* *rcv*,  
                  lbm\_uint32\_t *channel*, lbm\_rcv\_cb\_proc *proc*, void \* *clientd*)**

The callback *proc* will be called to deliver messages sent with the specified *channel* number. If NULL is specified for the *proc*, messages with the specified *channel* number will be delivered to the receiver's normal callback. If NULL is specified for the *proc*, any argument passed in for *clientd* will be ignored.

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*rcv* A pointer to a UM receiver object.  
*channel* A channel number to subscribe to.  
*proc* Pointer to a function to call when messages arrive.  
*clientd* Pointer to clientd data that is passed when data arrives and *proc* is called.

**Returns:**

0 for Success and -1 for Failure

**8.1.4.192 LBMEExpDLL int lbm\_rcv\_topic\_attr\_create (lbm\_rcv\_topic\_attr\_t  
                                  \*\* *attr*)**

The attribute object is allocated and filled with the current default values that are used by *lbm\_topic\_t* objects that concern receivers and may have been modified by a previously loaded configuration file.

**Parameters:**

*attr* A pointer to a pointer to a UM receiver topic attribute structure. Will be filled in by this function to point to the newly created *lbm\_rcv\_topic\_attr\_t* object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.193 LBMEExpDLL int lbm\_rcv\_topic\_attr\_create\_default  
(lbm\_rcv\_topic\_attr\_t \*\*attr)**

The attribute object is allocated and filled with the initial or factory default values built into LBM that are used by lbm\_topic\_t objects that concern receivers.

**Parameters:**

*attr* A pointer to a pointer to a UM receiver topic attribute structure. Will be filled in by this function to point to the newly created lbm\_rcv\_topic\_attr\_t object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.194 LBMEExpDLL int lbm\_rcv\_topic\_attr\_create\_from\_xml  
(lbm\_rcv\_topic\_attr\_t \*\*attr, const char \* context\_name, const char \* topicname)**

The attribute object is allocated and filled with the current default values that are used by lbm\_topic\_t objects that concern receivers and may have been modified by a previously loaded configuration file. If an XML configuration file has been loaded, the attribute object is further filled with the defaults for the given context name and receiver topic name. If the context name or receiver topic name are not permitted by the XML configuration, -1 is returned and no attribute object is created.

**Parameters:**

*attr* A pointer to a pointer to a UM receiver topic attribute structure. Will be filled in by this function to point to the newly created lbm\_rcv\_topic\_attr\_t object.

*context\_name* The context name used to lookup the receiver topic in the XML configuration. A NULL value is permitted, and will match unnamed contexts defined in the XML.

*topicname* The topic name used to lookup the receiver topic in the XML configuration. A NULL value is \*not\* permitted and will result in an error.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.195 LBMEExpDLL int lbm\_rcv\_topic\_attr\_delete (lbm\_rcv\_topic\_attr\_t \* attr)**

The attribute object is cleaned up and deleted.

**Parameters:**

*attr* Pointer to a UM source topic attribute object as returned by [lbm\\_rcv\\_topic\\_attr\\_create](#).

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.196 LBMEExpDLL int lbm\_rcv\_topic\_attr\_dump (lbm\_rcv\_topic\_attr\_t \* rattr, int \* size, lbm\_config\_option\_t \* opts)**

The config object is filled with source topic configuration options

**Parameters:**

*rcv* The receiver topic attribute object to retrieve the attributes from

*size* Size of the opts array. Will return the number of items that were set in opts

*opts* The options array to fill

**8.1.4.197 LBMEExpDLL int lbm\_rcv\_topic\_attr\_dup (lbm\_rcv\_topic\_attr\_t \*\* attr, const lbm\_rcv\_topic\_attr\_t \* original)**

A new attribute object is created as a copy of an existing object.

**Parameters:**

*attr* A pointer to a pointer to a UM receiver topic attribute structure. Will be filled in by this function to point to the newly created lbm\_rcv\_topic\_attr\_t object.

*original* Pointer to a UM receiver topic attribute object as returned by [lbm\\_rcv\\_topic\\_attr\\_create](#) or [lbm\\_rcv\\_topic\\_attr\\_create\\_default](#), from which *attr* is initialized.

**8.1.4.198 LBMEExpDLL int lbm\_rcv\_topic\_attr\_getopt (lbm\_rcv\_topic\_attr\_t \* attr, const char \* optname, void \* optval, size\_t \* optlen)****Parameters:**

*attr* Pointer to a UM receiver topic attribute object where the option is stored

*optname* String containing the option name

*optval* Pointer to the option value structure. The structure of the option values are specific to the options themselves.

*optlen* When passed, this is the max length (in bytes) of the *optval* structure.  
When returned, this is the length of the filled in *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

#### 8.1.4.199 LBMEExpDLL int lbm\_rev\_topic\_attr\_option\_size ()

The function returns the number of entries that are of type "source topic"

**Returns:**

The number of entries that are of type "source topic"

#### 8.1.4.200 LBMEExpDLL int lbm\_rev\_topic\_attr\_set\_from\_xml (*lbm\_rev\_topic\_attr\_t* \* *attr*, *const char* \* *context\_name*, *const char* \* *topicname*)

The attribute object is filled with the defaults for the given context name and receiver topic name, if an XML configuration file has been loaded. If the context name or receiver topic name are not permitted by the XML configuration, -1 is returned and the attribute object is not written to.

**Parameters:**

*attr* A pointer to a UM receiver topic attribute structure.

*context\_name* The context name used to lookup the receiver topic in the XML configuration. A NULL value is permitted, and will match unnamed contexts defined in the XML.

*topicname* The topic name used to lookup the receiver topic in the XML configuration. A NULL value is \*not\* permitted and will result in an error.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.201 LBMEExpDLL int lbm\_rcv\_topic\_attr\_setopt (lbm\_rcv\_topic\_attr\_t \*attr, const char \* optname, const void \* optval, size\_t optlen)**

Used before the topic is looked up and the receiver created. NOTE: the attribute object must first be initialized with the corresponding \_attr\_create() function.

**Parameters:**

- attr* Pointer to a UM receiver topic attribute object where the option is to be set.
- optname* String containing the option name.
- optval* Pointer to the option value structure. The structure of the option values are specific to the options themselves.
- optlen* Length (in bytes) of the *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.202 LBMEExpDLL int lbm\_rcv\_topic\_attr\_str\_getopt (lbm\_rcv\_topic\_attr\_t \* attr, const char \* optname, char \* optval, size\_t \* optlen)****Parameters:**

- attr* Pointer to a UM receiver topic attribute object where the option is stored
- optname* String containing the option name
- optval* String to be filled in with the option value.
- optlen* When passed, this is the max length (in bytes) of the string. When returned, this is the length of the filled in option value.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.203 LBMEExpDLL int lbm\_rcv\_topic\_attr\_str\_setopt (lbm\_rcv\_topic\_attr\_t \* attr, const char \* optname, const char \* optval)**

Used before the topic is looked up and the receiver created. NOTE: the attribute object must first be initialized with the corresponding \_attr\_create() function.

**Parameters:**

- attr* Pointer to a UM receiver topic attribute object where the option is to be set.

*optname* String containing the option name.

*optval* String containing the option value. The format of the string is specific to the options themselves.

**Returns:**

0 for Success and -1 for Failure.

#### 8.1.4.204 LBMExpDLL int lbt\_rcv\_topic\_dump (lbt\_rcv\_t \* *rcv*, int \* *size*, lbt\_config\_option\_t \* *opts*)

The config object is filled with source topic configuration options

**Parameters:**

*rcv* The receiver object to retrieve the attributes from

*size* Size of the opts array. Will return the number of items that were set in opts

*opts* The options array to fill

#### 8.1.4.205 LBMExpDLL int lbt\_rcv\_topic\_lookup (lbt\_topic\_t \*\* *topicp*, lbt\_context\_t \* *ctx*, const char \* *symbol*, const lbt\_rcv\_topic\_attr\_t \* *attr*)

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*topicp* A pointer to a pointer to a UM topic object. Will be filled in by this function to point to an lbt\_topic\_t object.

NOTE: Topic objects are cached. If a previously created topic object is found, it will be returned instead of a new object.

**Parameters:**

*ctx* Context object for topic

*symbol* The topic string. Topic strings should be limited in length to 246 characters (not including the final null).

*attr* Pointer to a receiver topic attributes object for passing in options

**Note:**

Setting attributes is only possible when a topic object is first created. This parameter will be ignored on subsequent lookups.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.206 LBMEExpDLL int lbm\_recv\_ume\_deregister (lbm\_recv\_t \* *rcv*)**

This function causes a UMP deregistration request to be sent to all stores the receiver is currently registered to, and disallows any future registrations.

**Parameters:**

*rcv* Pointer to an UM receiver object

**Returns:**

0 for Success and -1 for Failure

**8.1.4.207 LBMEExpDLL int lbm\_recv\_umq\_deregister (lbm\_recv\_t \* *rcv*, const char \* *queue\_name*)****Parameters:**

*rcv* Pointer to receiver object to de-register.

*queue\_name* Name of the queue or ULB source to deregister from. A NULL means de-register from all UMQ queues and ULB sources.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.208 LBMEExpDLL int lbm\_recv\_umq\_index\_release (lbm\_recv\_t \* *rcv*, const char \* *queue\_name*, lbm\_umq\_index\_info\_t \* *index\_info*)**

This function causes the UMQ indices to be assigned to another receiver.

**Parameters:**

*rcv* Pointer to receiver object that wishes to release the index.

*queue\_name* Name of the queue to reassign the index. A NULL means reassign the index for all UMQ queues.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.209 LBMEExpDLL int lmb\_rcv\_umq\_index\_reserve (lmb\_rcv\_t \* *rcv*,  
const char \* *queue\_name*, lmb\_umq\_index\_info\_t \* *index\_info*)**

This function causes the UMQ queue(s) or ULB sources to assign the specified index to this receiver if the queue ever happens to see a message sent on the specified index. If the index is already assigned, an error is returned as a LBM\_MSG\_UMQ\_INDEX\_ASSIGNMENT\_ERROR receiver event. Otherwise, if the reservation is successful, the receiver will get a LBM\_MSG\_UMQ\_INDEX\_ASSIGNED\_EX event.

**Parameters:**

- rcv* Pointer to receiver object that wishes to release the index.
- queue\_name* Name of the queue(s) at which to reserve the index. A NULL means reassign the index for all UMQ queues.
- index\_info* Index to reserve, or NULL to reserve a random unused numeric index.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.210 LBMEExpDLL int lmb\_rcv\_umq\_index\_start\_assignment (lmb\_rcv\_t  
\* *rcv*, const char \* *queue\_name*)****Parameters:**

- rcv* Pointer to receiver object to start assignment for.
- queue\_name* Name of the queue to start assignment from. A NULL means start assignment from all UMQ queues.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.211 LBMEExpDLL int lmb\_rcv\_umq\_index\_stop\_assignment (lmb\_rcv\_t  
\* *rcv*, const char \* *queue\_name*)**

This function causes new UMQ indices to not be assigned to the given receiver from the given UMQ queue(s). Messages with previously assigned UMQ indices may continue to be delivered to the given receiver from the given UMQ queue(s).

**Parameters:**

- rcv* Pointer to receiver object to stop assignment for.

***queue\_name*** Name of the queue to stop assignment from. A NULL means stop assignment from all UMQ queues.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.212 LBMEExpDLL int lbm\_recv\_umq\_queue\_msg\_list (lbm\_recv\_t \* *recv*,  
const char \* *queue\_name*, lbm\_umq\_msg\_selector\_t \* *selector*,  
lbm\_async\_operation\_func\_t \* *async\_opfunc*)**

Results are valid once the asynchronous operation callback is called with the LBM\_ASYNC\_OP\_STATUS\_COMPLETE status. An array of *lbm\_umq\_queue\_msg\_status\_t* objects is returned, each with its msgid field set to a valid UMQ message ID of a message that is currently in the queue within the application set to which the observer receiver belongs. All message IDs of all currently enqueued messages are returned. NOTE: The only valid field of each *lbm\_umq\_queue\_msg\_status\_t* object will be the msgid field; all other fields will be NULL or 0 and should not be relied upon to be accurate.

**Parameters:**

*recv* Pointer to observer receiver object

***queue\_name*** Name of the queue to retrieve the list of messages from.

***selector*** Not currently supported; please pass NULL.

***async\_opfunc*** The asynchronous operation callback the topic list will be delivered to.

**8.1.4.213 LBMEExpDLL int lbm\_recv\_umq\_queue\_msg\_retrieve (lbm\_recv\_t  
\* *recv*, const char \* *queue\_name*, lbm\_umq\_msgid\_t \* *msgids*, int  
*num\_msgids*, lbm\_async\_operation\_func\_t \* *async\_opfunc*)**

Results are valid once the asynchronous operation callback is called with the LBM\_ASYNC\_OP\_STATUS\_COMPLETE status. An array of *lbm\_umq\_queue\_msg\_status\_t* objects is returned; the size of the array returned will match the size of the msgids array originally passed in. Each returned *lbm\_umq\_queue\_msg\_status\_t* object contains state information (LBM\_UMQ\_QUEUE\_MSG\_STATUS\_UNASSIGNED, LBM\_UMQ\_QUEUE\_MSG\_STATUS\_CONSUMED, etc.), and possibly message data, for each requested message. If message data was available, the msg field of the *lbm\_umq\_queue\_msg\_status\_t* object will be non-NULL; otherwise it will be NULL. Message data is not always still available for a given message ID (in the case of a message that has been fully consumed across all configured application sets in the queue, for example).

See also:

[LBM\\_UMQ\\_QUEUE\\_MSG\\_STATUS\\_UNKNOWN](#)  
[LBM\\_UMQ\\_QUEUE\\_MSG\\_STATUS\\_UNASSIGNED](#)  
[LBM\\_UMQ\\_QUEUE\\_MSG\\_STATUS\\_ASSIGNED](#)  
[LBM\\_UMQ\\_QUEUE\\_MSG\\_STATUS\\_REASSIGNING](#)  
[LBM\\_UMQ\\_QUEUE\\_MSG\\_STATUS\\_CONSUMED](#)

Parameters:

*rcv* Pointer to observer receiver object  
*queue\_name* Name of the queue to retrieve the messages from.  
*msgids* Array of UMQ message IDs to retrieve.  
*num\_msgids* Length of message ID array.  
*async\_opfunc* The asynchronous operation callback the retrieved messages will be delivered to.

#### 8.1.4.214 LBMEExpDLL int **lbt\_rcv\_unsubscribe\_channel** (*lbt\_rcv\_t \* rcv, lbt\_uint32\_t channel*)

Remove a subscription to a channel previously subscribed to with

See also:

[lbt\\_rcv\\_subscribe\\_channel](#).

Parameters:

*rcv* A pointer to a UM receiver object.  
*channel* The channel number for the channel subscription to be removed.

Returns:

0 for Success and -1 for Failure

#### 8.1.4.215 LBMEExpDLL int **lbt\_rcv\_unsubscribe\_channel\_ex** (*lbt\_rcv\_t \* rcv, lbt\_uint32\_t channel, lbt\_event\_queue\_cancel\_cb\_info\_t \* cbinfo*)

Parameters:

*rcv* A pointer to a UM receiver object.  
*channel* The channel number for the channel subscription to be removed.  
*cbinfo* Cancellation callback information, containing the event queue, function pointer, and client data for the callback.

**Returns:**

0 for Success and -1 for Failure

**8.1.4.216 LBMEExpDLL int lbm\_register\_fd (lbm\_context\_t \* *ctx*, lbm\_handle\_t *handle*, lbm\_fd\_cb\_proc *proc*, void \* *clientd*, lbm\_event\_queue\_t \* *evq*, lbm\_ulong\_t *ev*)**

This registers a file descriptor/socket event that will call the function *proc* at a later time passing in specified data when the event occurs. NOTE: this functionality is not available for Windows-based contexts where completion ports are specified. See configuration option "context fd\_management\_type".

**See also:**

[lbm\\_cancel\\_fd](#)

**Warning:**

It is not recommended to call this function from a context thread callback.

**Parameters:**

*ctx* Pointer to the UM context object

*handle* file descriptor/socket of interest for event.

*proc* Pointer to the function to call when the event occurs

*clientd* Pointer to client data that is passed when the event occurs.

*evq* Optional Event Queue to place events on when they occur. If NULL causes *proc* to be called from context thread.

*ev* One or more of LBM\_FD\_EVENT\_\*(ORed to together). Mask of events of interest.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.217 LBMEExpDLL int lbm\_request\_delete (lbm\_request\_t \* *req*)**

Delete a UM request object. When this function is used, subsequent responses for this given request object will be ignored. Note that this function can return while the callback may still be executing if request events are being delivered via an event queue. If the application needs to know when all possible processing on the request is complete, it must use [lbm\\_request\\_delete\\_ex\(\)](#).

**Parameters:**

*req* A pointer to an `lbt_request_t` object returned by `lbt_send_request`.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.218 LBMEExpDLL int lbt\_request\_delete\_ex (lbt\_request\_t \* *req*,  
lbt\_event\_queue\_cancel\_cb\_info\_t \* *cbinfo*)**

Delete a UM request object, with an application callback indicating when the request is fully canceled. When this function is used, any more responses for this given request object will be ignored. This extended version of the request cancel function requires the configuration option `queue_cancellation_callbacks_enabled` be set to 1.

**Parameters:**

*req* A pointer to an `lbt_request_t` object returned by `lbt_send_request`.

*cbinfo* Cancellation callback information, containing the event queue, function pointer, and client data for the callback.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.219 LBMEExpDLL int lbt\_response\_delete (lbt\_response\_t \* *resp*)**

When an application receives a *request*, the message object `lbt_msg_t` contains a response object *response* which is used by `lbt_send_response()`. Normally, when the receive callback returns, both the message and the response object are deleted by UM. However the receive callback can optionally save the response object and set `msg->response=NULL` to prevent UM from deleting it when the receive callback returns. It then becomes the application's responsibility to delete the response when appropriate.

**See also:**

`lbt_msg_t`, [lbt\\_msg\\_delete](#)

**Parameters:**

*resp* A pointer to an `lbt_response_t` object given in a `lbt_msg_t` object.

**Returns:**

0 for Success and -1 for Failure.

---

**8.1.4.220 LBMExpDLL int lbm\_schedule\_timer (lbm\_context\_t \* *ctx*,  
          lbm\_timer\_cb\_proc *proc*, void \* *clientd*, lbm\_event\_queue\_t \* *evq*,  
          lbm\_ulong\_t *delay*)**

This schedules a timer that will call the function *proc* at a later time passing in specified data. This is a one-shot timer. To implement a recurring timer, the callback function should call [lbm\\_schedule\\_timer\(\)](#) again.

A zero duration timer is legal and causes the associated callback to be called as soon as possible on the context thread or to be enqueued as an event on the associated event queue. In this case, the event queue dispatching thread calls the associated callback after all currently pending events have been dispatched.

**See also:**

[lbm\\_cancel\\_timer](#)

**Parameters:**

*ctx* Pointer to the UM context object

*proc* Pointer to the function to call when the timer expires.

*clientd* Pointer to client data that is passed when the timer expires.

*evq* Optional Event Queue to place timer events on when they occur. If NULL causes *proc* to be called from context thread.

*delay* Delay until *proc* should be called (in milliseconds).

**Returns:**

An identifier for the timer that may be used to cancel it or -1 for Failure.

---

**8.1.4.221 LBMExpDLL int lbm\_schedule\_timer\_recurring (lbm\_context\_t \*  
                  *ctx*, lbm\_timer\_cb\_proc *proc*, void \* *clientd*, lbm\_event\_queue\_t \* *evq*,  
                  lbm\_ulong\_t *delay*)**

This schedules a recurring timer that calls the function *proc* at the given time interval passing in the specified data. The timer will reschedule itself each time it expires and must be explicitly canceled to stop the timer. Caution should be exercised with the delay since timer events will build up if the application can not process the events fast enough.

**See also:**

[lbm\\_cancel\\_timer](#)

**Parameters:**

*ctx* Pointer to the UM context object

***proc*** Pointer to the function to call when the timer expires.

***clientd*** Pointer to client data that is passed when the timer expires.

***evq*** Optional Event Queue to place timer events on when they occur. If NULL causes *proc* to be called from context thread.

***delay*** Delay until *proc* should be called (in milliseconds).

**Returns:**

An identifier for the timer that may be used to cancel it or -1 for Failure.

#### 8.1.4.222 LBMExpDLL int **lbt\_send\_request** (**lbt\_request\_t** \*\**reqp*, **lbt\_src\_t** \**src*, const char \**data*, size\_t *len*, **lbt\_request\_cb\_proc** *proc*, void \**clientd*, **lbt\_event\_queue\_t** \**evq*, int *send\_flags*)

This function creates a request object *reqp* which is used by UM to route responses to the desired application callback *proc* and must be retained until all responses are received. When the requestor does not expect any additional responses, it deletes the request object using [lbt\\_request\\_delete\(\)](#).

**See also:**

[lbt\\_src\\_send](#)

**Warning:**

It is not recommended to call this function from a context thread callback. If called from a context thread callback, use the LBM\_SRC\_NONBLOCK flag and handle any LBM\_EWOULDBLOCK errors internally.

**Parameters:**

***reqp*** A pointer to a pointer for the **lbt\_request\_t** object created to be stored.

***src*** The UM source to send the request out.

***data*** Buffer to be included as data in the request.

***len*** Length (in bytes) of the data included with the request.

***proc*** Pointer to function to call when responses come in for this request.

***clientd*** Client data returned in the callback proc *proc*.

***evq*** Optional Event Queue to place message events on when they occur. If NULL causes *proc* to be called from context thread.

***send\_flags*** Flags used to instruct UM how to handle this message. See [lbt\\_src\\_send\(\)](#) for more information.

**Returns:**

0 for Success and -1 for Failure.

---

**8.1.4.223 LBMExpDLL int lbm\_send\_request\_ex (lbm\_request\_t \*\**reqp*,  
          lbm\_src\_t \**src*, const char \**data*, size\_t *len*, lbm\_request\_cb\_proc  
          *proc*, void \**clientd*, lbm\_event\_queue\_t \**evq*, int *send\_flags*,  
          lbm\_src\_send\_ex\_info\_t \**exinfo*)**

This function creates a request object *reqp* which is used by UM to route responses to the desired application callback *proc* and must be retained until all responses are received. When the requestor does not expect any additional responses, it deletes the request object using [lbm\\_request\\_delete\(\)](#).

**See also:**

[lbm\\_src\\_send](#)

**Warning:**

It is not recommended to call this function from a context thread callback. If called from a context thread callback, use the LBM\_SRC\_NONBLOCK flag and handle any LBM\_EWOULDBLOCK errors internally.

**Parameters:**

*reqp* A pointer to a pointer for the lbm\_request\_t object created to be stored.  
*src* The UM source to send the request out.  
*data* Buffer to be included as data in the request.  
*len* Length (in bytes) of the data included with the request.  
*proc* Pointer to function to call when responses come in for this request.  
*clientd* Client data returned in the callback proc *proc*.  
*evq* Optional Event Queue to place message events on when they occur. If NULL causes *proc* to be called from context thread.  
*send\_flags* Flags used to instruct UM how to handle this message. See [lbm\\_src\\_send\(\)](#) for more information.  
*exinfo* Pointer to lbm\_src\_send\_ex\_info\_t options

**Returns:**

0 for Success and -1 for Failure.

---

**8.1.4.224 LBMExpDLL int lbm\_send\_response (lbm\_response\_t \**resp*, const  
          char \**data*, size\_t *len*, int *flags*)**

**Parameters:**

*resp* A pointer to an lbm\_response\_t object given in a lbm\_msg\_t object.

***data*** Buffer to send as the response data.

***len*** Length (in bytes) of the data to send as the response data.

***flags*** Flags indicating various conditions. ORed set of values.

- `LBM_SRC_NONBLOCK` - If messages could not be sent immediately return and error and signal `LBM_EWOULDBLOCK`.
- `LBM_SRC_BLOCK` - Block the caller indefinitely until the message is sent. (This behavior is the default if neither `LBM_SRC_NONBLOCK` nor `LBM_SRC_BLOCK` are supplied.)

**Returns:**

-1 for Failure or the number of bytes sent if Successful.

#### 8.1.4.225 LBMEExpDLL `lbm_serialized_response_t*` `lbm_serialize_response` (`lbm_response_t *resp`)

An UM response object (`lbm_response_t`) may be serialized to allow applications other than the one originally receiving a request to respond to that request.

**Parameters:**

***resp*** A pointer to an `lbm_response_t` object.

**Returns:**

A pointer to an `lbm_serialized_response_t` object.

#### 8.1.4.226 LBMEExpDLL `int` `lbm_serialized_response_delete` (`lbm_serialized_response_t *serialized_response`)

After a serialized UM response object has been copied or used, it is the application's responsibility to delete the serialized response object.

**Parameters:**

***serialized\_response*** A pointer to an `lbm_serialized_response_t` object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.227 LBMEExpDLL void lbm\_set\_lbtrm\_loss\_rate (int *rate*)**

Set the LBT-RM loss rate. This is equivalent to setting the environment variable LBTRM\_LOSS\_RATE, but allows the loss rate to be changed under program control.

**Parameters:**

*rate* Loss rate (from 0 to 100).

**8.1.4.228 LBMEExpDLL void lbm\_set\_lbtrm\_src\_loss\_rate (int *rate*)**

Set the LBT-RM source loss rate. This is equivalent to setting the environment variable LBTRM\_SRC\_LOSS\_RATE, but allows the loss rate to be changed under program control.

**Parameters:**

*rate* Loss rate (from 0 to 100).

**8.1.4.229 LBMEExpDLL void lbm\_set\_lbtru\_loss\_rate (int *rate*)**

Set the LBT-RU loss rate. This is equivalent to setting the environment variable LBTRU\_LOSS\_RATE, but allows the loss rate to be changed under program control.

**Parameters:**

*rate* Loss rate (from 0 to 100).

**8.1.4.230 LBMEExpDLL void lbm\_set\_lbtru\_src\_loss\_rate (int *rate*)**

Set the LBT-RU source loss rate. This is equivalent to setting the environment variable LBTRU\_SRC\_LOSS\_RATE, but allows the loss rate to be changed under program control.

**Parameters:**

*rate* Loss rate (from 0 to 100).

**8.1.4.231 LBMEExpDLL int lbm\_set\_umm\_info (lbm\_umm\_info\_t \* *info*)**

In order to be effective, this function \*must\* be called before any other LBM API function.

See also:

[lbm\\_umm\\_info\\_t](#)

Parameters:

*info* lbm\_umm\_info\_t struct specifying options for connecting to a UMM daemon.

Returns:

0 for Success and -1 for Failure.

#### 8.1.4.232 LBMEExpDLL int lbm\_src\_channel\_create (lbm\_src\_channel\_info\_t \*\**chnp*, lbm\_src\_t \**src*, lbm\_uint32\_t *channel\_num*)

Parameters:

*chnp* A pointer to a pointer to a UM channel info object. Will be filled in by this function to point to the newly created lbm\_src\_channel\_info\_t object.

*src* Pointer to the UM source the channel info object will be used with.

*channel\_num* A channel number in the range 0-4294967295

See also:

[lbm\\_src\\_send\\_ex](#) [lbm\\_src\\_sendv\\_ex](#)

Returns:

0 for Success and -1 for Failure

#### 8.1.4.233 LBMEExpDLL int lbm\_src\_channel\_delete (lbm\_src\_channel\_info\_t \**chn*)

Parameters:

*chn* A pointer to a channel info object allocated with lbm\_src\_channel\_create.

Returns:

0 for Success and -1 for Failure

**8.1.4.234 LBMExpDLL int lbm\_src\_create (lbm\_src\_t \*\**srcp*, lbm\_context\_t \**ctx*, lbm\_topic\_t \**topic*, lbm\_src\_cb\_proc *proc*, void \**clientd*, lbm\_event\_queue\_t \**evq*)**

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*srcp* A pointer to a pointer to a UM source object. Will be filled in by this function to point to the newly created lbm\_src\_t object.

*ctx* Pointer to the UM context object associated with the sender.

*topic* Pointer to the UM topic object associated with the destination of messages sent by the source.

*proc* Pointer to a function to call when events occur related to the source. If NULL, then events are not delivered to the source.

*clientd* Pointer to client data that is passed when *proc* is called.

*evq* Optional Event Queue to place events on when they occur. If NULL causes *proc* to be called from context thread.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.235 LBMExpDLL int lbm\_src\_delete (lbm\_src\_t \**src*)**

Delete a UM source object. Note that this function can return while the source callback may still be executing if source events are being delivered via an event queue. If the application needs to know when all possible processing on the source is complete, it must use [lbm\\_src\\_delete\\_ex\(\)](#).

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*src* Pointer to a UM source object to delete.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.236 LBMEExpDLL int lbm\_src\_delete\_ex (lbm\_src\_t \* src,  
                          lbm\_event\_queue\_cancel\_cb\_info\_t \* cbinfo)**

Delete a UM source object with an application callback indicating when the source is fully canceled. This extended version of the source delete function requires the configuration option queue\_cancellation\_callbacks\_enabled be set to 1.

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*src* Pointer to a UM source object to delete.

*cbinfo* Cancellation callback information, containing the event queue, function pointer, and client data for the callback.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.237 LBMEExpDLL int lbm\_src\_flush (lbm\_src\_t \* src)****Warning:**

Calling this function from a context thread callback for stability and confirmation events could cause a deadlock

**Parameters:**

*src* Pointer to the UM source to send from

**Returns:**

-1 for Failure or 0 for Success.

**8.1.4.238 LBMEExpDLL int lbm\_src\_get\_inflight (lbm\_src\_t \* src, int type, int \*  
                          inflight, lbm\_flight\_size\_set\_inflight\_cb\_proc proc, void \* clientd)****See also:**

[lbm\\_flight\\_size\\_set\\_inflight\\_cb\\_proc](#)

**Parameters:**

*src* Pointer to the source.

*type* Type of flight size.

- LBM\_FLIGHT\_SIZE\_TYPE\_UME - Specifies a UM flight size
- LBM\_FLIGHT\_SIZE\_TYPE\_ULB - Specifies a ULB flight size
- LBM\_FLIGHT\_SIZE\_TYPE\_UMQ - Specifies a UMQ flight size

*inflight* Pointer to an int whose value will be filled in to reflect the current inflight.

*proc* Optional callback that allows an application to set the current inflight.

*clientd* Optional client data passed into the proc.

#### Returns:

0 for Success, -1 for failure if the proc returns a negative value.

**8.1.4.239** **LBMExpDLL int lbm\_src\_get\_inflight\_ex (lbm\_src\_t \* src, int type, lbm\_flight\_size\_inflight\_t \* inflight, lbm\_flight\_size\_set\_inflight\_ex\_cb\_proc proc, void \* clientd)**

#### See also:

[lbm\\_flight\\_size\\_set\\_inflight\\_cb\\_proc](#)

#### Parameters:

*src* Pointer to the source.

*type* Type of flight size.

- LBM\_FLIGHT\_SIZE\_TYPE\_UME - Specifies a UM flight size
- LBM\_FLIGHT\_SIZE\_TYPE\_ULB - Specifies a ULB flight size
- LBM\_FLIGHT\_SIZE\_TYPE\_UMQ - Specifies a UMQ flight size

*inflight* Pointer to a structure to be filled in with the current inflight values.

*proc* Optional callback that allows an application to set the current inflight.

*clientd* Optional client data passed into the proc.

#### Returns:

0 for Success, -1 for failure if the proc returns a negative value.

**8.1.4.240** **LBMExpDLL int lbm\_src\_getopt (lbm\_src\_t \* src, const char \* optname, void \* optval, size\_t \* optlen)**

#### Parameters:

*src* Pointer to a UM source object where the option is stored

*optname* String containing the option name

*optval* Pointer to the option value structure. The structure of the option values are specific to the options themselves.

*optlen* When passed, this is the max length (in bytes) of the *optval* structure.  
When returned, this is the length of the filled in *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

#### **8.1.4.241 LBMEExpDLL int lhm\_src\_reset\_transport\_stats (lhm\_src\_t \* src)**

**Parameters:**

*src* Pointer to the UM source to reset statistics for.

**Returns:**

-1 for Failure and 0 for Success.

#### **8.1.4.242 LBMEExpDLL int lhm\_src\_retrieve\_transport\_stats (lhm\_src\_t \* src, lhm\_src\_transport\_stats\_t \* stats)**

**Parameters:**

*src* Pointer to the UM source to retrieve statistics for.

*stats* Pointer to a stats structure to fill in.

**Returns:**

-1 for Failure and 0 for Success.

#### **8.1.4.243 LBMEExpDLL int lhm\_src\_send (lhm\_src\_t \* src, const char \* msg, size\_t len, int flags)**

**Warning:**

It is not recommended to call this function from a context thread callback. If called from a context thread callback, use the LBM\_SRC\_NONBLOCK flag and handle any LBM\_EWOULDBLOCK errors internally.

Calling this function from a context thread callback for stability and confirmation events could cause a deadlock.

**Parameters:**

***src*** Pointer to the UM source to send from

***msg*** Pointer to the data to send in this message

***len*** Length (in bytes) of the data to send in this message

***flags*** Flags indicating various conditions. ORed set of values.

- LBM\_MSG\_START\_BATCH - Message starts a batch of messages
- LBM\_MSG\_END\_BATCH - Message ends a batch of messages. Batch should be sent to the implicit batching buffer.
- LBM\_MSG\_COMPLETE\_BATCH - Message constitutes a complete batch and should be sent to the implicit batching buffer.
- LBM\_MSG\_FLUSH - Message is to be sent ASAP (not implicitly or explicitly batched). This also flushes waiting messages that were explicitly or implicitly batched.
- LBM\_SRC\_NONBLOCK - If message could not be sent immediately return and error and signal LBM\_EWOULDBLOCK.
- LBM\_SRC\_BLOCK - Block the caller indefinitely until the message is sent. (This behavior is the default if neither LBM\_SRC\_NONBLOCK nor LBM\_SRC\_BLOCK are supplied.)

**Returns:**

-1 for Failure or 0 for Success.

#### 8.1.4.244 LBMExpDLL int lbm\_src\_send\_ex (lbm\_src\_t \* src, const char \* msg, size\_t len, int flags, lbm\_src\_send\_ex\_info\_t \* info)

**Warning:**

If called from a context thread callback, use the LBM\_SRC\_NONBLOCK flag and handle any LBM\_EWOULDBLOCK errors internally.

Calling this function from a context thread callback for stability and confirmation events could cause a deadlock

**Parameters:**

***src*** Pointer to the UM source to send from

***msg*** Pointer to the data to send in this message

***len*** Length (in bytes) of the data to send in this message

***flags*** Flags indicating various conditions. ORed set of values.

- LBM\_MSG\_START\_BATCH - Message starts a batch of messages
- LBM\_MSG\_END\_BATCH - Message ends a batch of messages. Batch should be sent to the implicit batching buffer.

- LBM\_MSG\_COMPLETE\_BATCH - Message constitutes a complete batch and should be sent to the implicit batching buffer.
- LBM\_MSG\_FLUSH - Message is to be sent ASAP (not implicitly or explicitly batched). This also flushes waiting messages that were explicitly or implicitly batched.
- LBM\_SRC\_NONBLOCK - If message could not be sent immediately return and error and signal LBM\_EWOULDBLOCK.
- LBM\_SRC\_BLOCK - Block the caller indefinitely until the message is sent. (This behavior is the default if neither LBM\_SRC\_NONBLOCK nor LBM\_SRC\_BLOCK are supplied.)

**info** Pointer to `lbt_src_send_ex_info_t` options

**Returns:**

-1 for Failure or 0 for Success.

#### 8.1.4.245 `LBMExpDLL int lbt_src_sendv (lbt_src_t *src, const lbt_iovec_t *iov, int num, int flags)`

The messages are specified as an array of iovecs. Be aware that each iovec element is considered as a full application message unless LBM\_MSG iov\_GATHER is used in the flags field. In that case, the elements of the array are gathered together into a single message.

**Warning:**

It is not recommended to call this function from a context thread callback. If called from a context thread callback, use the LBM\_SRC\_NONBLOCK flag and handle any LBM\_EWOULDBLOCK errors internally.

**Parameters:**

**src** Pointer to the UM source to send from.

**iov** Pointer to an array of iovecs that hold message information.

**num** Number of elements of the iov array to send.

**flags** Flags indicating various conditions. ORed set of values.

- LBM\_MSG\_START\_BATCH - Messages start a batch of messages
- LBM\_MSG\_END\_BATCH - Messages end a batch of messages. Batch should be sent to the implicit batching buffer.
- LBM\_MSG\_COMPLETE\_BATCH - Messages constitute a complete batch and should be sent to the implicit batching buffer.
- LBM\_MSG\_FLUSH - Messages are to be sent ASAP (not implicitly batched or explicitly batched).

- LBM\_SRC\_NONBLOCK - If messages could not be sent immediately return and error and signal LBM\_EWOULDBLOCK.
- LBM\_SRC\_BLOCK - Block the caller indefinitely until the messages are all sent. (This behavior is the default if neither LBM\_SRC\_NONBLOCK nor LBM\_SRC\_BLOCK are supplied.)
- LBM\_MSG iov\_GATHER - iovec elements should be gather into a single message.

**Returns:**

-1 for Failure or 0 for Success.

#### **8.1.4.246 LBMExpDLL int lbm\_src\_sendv\_ex (lbm\_src\_t \* src, const                   lbm\_iovec\_t \* iov, int num, int flags, lbm\_src\_send\_ex\_info\_t \* info)**

The messages are specified as an array of iovecs. The elements of the array are gathered together into a single message.

**Warning:**

If called from a context thread callback, use the LBM\_SRC\_NONBLOCK flag and handle any LBM\_EWOULDBLOCK errors internally.

Calling this function from a context thread callback for stability and confirmation events could cause a deadlock

**Parameters:**

**src** Pointer to the UM source to send from

**iov** Pointer to an array of iovecs that hold message information.

**num** Number of elements of the iov array to send.

**flags** Flags indicating various conditions. ORed set of values.

- LBM\_MSG\_START\_BATCH - Message starts a batch of messages
- LBM\_MSG\_END\_BATCH - Message ends a batch of messages. Batch should be sent to the implicit batching buffer.
- LBM\_MSG\_COMPLETE\_BATCH - Message constitutes a complete batch and should be sent to the implicit batching buffer.
- LBM\_MSG\_FLUSH - Message is to be sent ASAP (not implicitly or explicitly batched). This also flushes waiting messages that were explicitly or implicitly batched.
- LBM\_SRC\_NONBLOCK - If message could not be sent immediately return and error and signal LBM\_EWOULDBLOCK.
- LBM\_SRC\_BLOCK - Block the caller indefinitely until the message is sent. (This behavior is the default if neither LBM\_SRC\_NONBLOCK nor LBM\_SRC\_BLOCK are supplied.)

*info* Pointer to lbm\_src\_send\_ex\_info\_t options

**Returns:**

-1 for Failure or 0 for Success.

#### 8.1.4.247 LBMEExpDLL int lbm\_src\_setopt (lbm\_src\_t \* *src*, const char \* *optname*, const void \* *optval*, size\_t *optlen*)

Only those options that can be set during operation may be specified. See "Options That May Be Set During Operation" (doc/Config/maybesetduringoperation.html) in The UM Configuration Guide. For API functions that can access any option, see lbm\_src\_topic\_attr\_\*().

**Parameters:**

*src* Pointer to a UM source object where the option is to be set

*optname* String containing the option name

*optval* Pointer to the option value structure. The structure of the option values are specific to the options themselves.

*optlen* Length (in bytes) of the *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

#### 8.1.4.248 LBMEExpDLL int lbm\_src\_str\_getopt (lbm\_src\_t \* *src*, const char \* *optname*, char \* *optval*, size\_t \* *optlen*)

**Parameters:**

*src* Pointer to a UM source object where the option is stored

*optname* String containing the option name

*optval* String to be filled in with the option value.

*optlen* When passed, this is the max length (in bytes) of the string. When returned, this is the length of the filled in option value.

**Returns:**

0 for Success and -1 for Failure.

---

**8.1.4.249 LBMEExpDLL int lbm\_src\_str\_setopt (lbm\_src\_t \* src, const char \* optname, const char \* optval)**

Only those options that can be set during operation may be specified. See "Options That May Be Set During Operation" (doc/Config/maybesetduringoperation.html) in The UM Configuration Guide. For API functions that can access any option, see `lbm_src_topic_attr_*`().

**Parameters:**

- src* Pointer to a UM source object where the option is to be set
- optname* String containing the option name
- optval* String containing the option value. The format of the string is specific to the options themselves.

**Returns:**

0 for Success and -1 for Failure.

---

**8.1.4.250 LBMEExpDLL int lbm\_src\_topic\_alloc (lbm\_topic\_t \*\* topicp, lbm\_context\_t \* ctx, const char \* symbol, const lbm\_src\_topic\_attr\_t \* attr)**

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

- topicp* A pointer to a pointer to a UM topic object. Will be filled in by this function to point to the newly created `lbm_topic_t` object.
- ctx* Context object for Topic
- symbol* The Topic string. Topic strings should be limited in length to 246 characters (not including the final null).
- attr* Pointer to a Src Topic attribute object for passing in options

**Returns:**

0 for Success and -1 for Failure.

---

**8.1.4.251 LBMEExpDLL int lbm\_src\_topic\_attr\_create (lbm\_src\_topic\_attr\_t \*\* attr)**

The attribute object is filled with the current default values that are used by `lbm_topic_t` objects that concern sources and may have been modified by a previously loaded configuration file.

**Parameters:**

***attr*** A pointer to a pointer to a UM source topic attribute structure. Will be filled in by this function to point to the newly created `lbt_src_topic_attr_t` object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.252 LBMEExpDLL int lbt\_src\_topic\_attr\_create\_default  
(lbt\_src\_topic\_attr\_t \*\* *attr*)**

The attribute object is allocated and filled with the initial or factory default values built into LBM that are used by `lbt_topic_t` objects that concern sources.

**Parameters:**

***attr*** A pointer to a pointer to a UM source topic attribute structure. Will be filled in by this function to point to the newly created `lbt_src_topic_attr_t` object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.253 LBMEExpDLL int lbt\_src\_topic\_attr\_create\_from\_xml  
(lbt\_src\_topic\_attr\_t \*\* *attr*, const char \* *context\_name*, const char \*  
*topicname*)**

The attribute object is allocated and filled with the current default values that are used by `lbt_topic_t` objects that concern sources and may have been modified by a previously loaded configuration file. If an XML configuration file has been loaded, the attribute object is further filled with the defaults for the given context name and source topic name. If the context name or source topic name are not permitted by the XML configuration, -1 is returned and no attribute object is created.

**Parameters:**

***attr*** A pointer to a pointer to a UM source topic attribute structure. Will be filled in by this function to point to the newly created `lbt_src_topic_attr_t` object.

***context\_name*** The context name used to lookup the source topic in the XML configuration. A NULL value is permitted, and will match unnamed contexts defined in the XML.

***topicname*** The topic name used to lookup the source topic in the XML configuration. A NULL value is \*not\* permitted and will result in an error.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.254 LBMEExpDLL int lbm\_src\_topic\_attr\_delete (lbm\_src\_topic\_attr\_t \* attr)**

The attribute object is cleaned up and deleted.

**Parameters:**

*attr* Pointer to a UM source topic attribute object as returned by [lbm\\_src\\_topic\\_attr\\_create](#).

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.255 LBMEExpDLL int lbm\_src\_topic\_attr\_dump (lbm\_src\_topic\_attr\_t \* sattr, int \* size, lbm\_config\_option\_t \* opts)**

The config object is filled with source topic configuration options

**Parameters:**

*src* The source topic attribute object to retrieve the attributes from

*size* Size of the opts array. Will return the number of items that were set in opts

*opts* The options array to fill

**8.1.4.256 LBMEExpDLL int lbm\_src\_topic\_attr\_dup (lbm\_src\_topic\_attr\_t \*\* attr, const lbm\_src\_topic\_attr\_t \* original)**

A new attribute object is created as a copy of an existing object.

**Parameters:**

*attr* A pointer to a pointer to a UM source topic attribute structure. Will be filled in by this function to point to the newly created lbm\_src\_topic\_attr\_t object.

*original* Pointer to a UM source topic attribute object as returned by [lbm\\_src\\_topic\\_attr\\_create](#) or [lbm\\_src\\_topic\\_attr\\_create\\_default](#), from which *attr* is initialized.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.257 LBMEExpDLL int lbm\_src\_topic\_attr\_getopt (lbm\_src\_topic\_attr\_t \*  
attr, const char \* optname, void \* optval, size\_t \* optlen)**

**Parameters:**

*attr* Pointer to a UM source topic attribute object where the option is stored  
*optname* String containing the option name  
*optval* Pointer to the option value structure. The structure of the option values are specific to the options themselves.  
*optlen* When passed, this is the max length (in bytes) of the *optval* structure.  
When returned, this is the length of the filled in *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.258 LBMEExpDLL int lbm\_src\_topic\_attr\_option\_size ()**

The function returns the number of entries that are of type "topic"

**Returns:**

The number of entries that are of type "topic"

**8.1.4.259 LBMEExpDLL int lbm\_src\_topic\_attr\_set\_from\_xml  
(lbm\_src\_topic\_attr\_t \* attr, const char \* context\_name, const char \*  
topicname)**

The attribute object is filled with the defaults for the given context name and source topic name, if an XML configuration file has been loaded. If the context name or source topic name are not permitted by the XML configuration, -1 is returned and the attribute object is not written to.

**Parameters:**

*attr* A pointer to a UM source topic attribute structure.  
*context\_name* The context name used to lookup the source topic in the XML configuration. A NULL value is permitted, and will match unnamed contexts defined in the XML.  
*topicname* The topic name used to lookup the source topic in the XML configuration. A NULL value is \*not\* permitted and will result in an error.

**Returns:**

0 for Success and -1 for Failure.

---

**8.1.4.260 LBMExpDLL int lbm\_src\_topic\_attr\_setopt (lbm\_src\_topic\_attr\_t \* attr, const char \* optname, const void \* optval, size\_t optlen)**

Used before the topic is allocated and the source created. NOTE: the attribute object must first be initialized with the corresponding \_attr\_create() function.

**Parameters:**

*attr* Pointer to a UM source topic attribute object where the option is to be set  
*optname* String containing the option name  
*optval* Pointer to the option value structure. The structure of the option values are specific to the options themselves.  
*optlen* Length (in bytes) of the *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

---

**8.1.4.261 LBMExpDLL int lbm\_src\_topic\_attr\_str\_getopt (lbm\_src\_topic\_attr\_t \* attr, const char \* optname, char \* optval, size\_t \* optlen)**

**Parameters:**

*attr* Pointer to a UM source topic attribute object where the option is stored  
*optname* String containing the option name  
*optval* String to be filled in with the option value.  
*optlen* When passed, this is the max length (in bytes) of the string. When returned, this is the length of the filled in option value.

**Returns:**

0 for Success and -1 for Failure.

---

**8.1.4.262 LBMExpDLL int lbm\_src\_topic\_attr\_str\_setopt (lbm\_src\_topic\_attr\_t \* attr, const char \* optname, const char \* optval)**

Used before the topic is allocated and the source created. NOTE: the attribute object must first be initialized with the corresponding \_attr\_create() function.

**Parameters:**

*attr* Pointer to a UM source topic attribute object where the option is to be set

*optname* String containing the option name

*optval* String containing the option value. The format of the string is specific to the options themselves.

**Returns:**

0 for Success and -1 for Failure.

#### 8.1.4.263 LBMExpDLL int lbm\_src\_topic\_dump (lbm\_src\_t \* src, int \* size,                           lbm\_config\_option\_t \* opts)

The config object is filled with source topic configuration options

**Parameters:**

*src* The source object to retrieve the attributes from

*size* Size of the opts array. Will return the number of items that were set in opts

*opts* The options array to fill

#### 8.1.4.264 LBMExpDLL int lbm\_src\_ume\_deregister (lbm\_src\_t \* src)

This function causes a UMP deregistration to be sent to all stores the source is currently registered to, and disallows any future registrations from taking place.

**Parameters:**

*src* Pointer to a UM source object

**Returns:**

0 for Success and -1 for Failure.

#### 8.1.4.265 LBMExpDLL lbm\_topic\_t\* lbm\_topic\_from\_src (lbm\_src\_t \* src)

**Parameters:**

*src* Pointer to a UM source object.

**Returns:**

A pointer to the UM topic object associated with the UM source object.

**8.1.4.266 LBMEExpDLL int lbm\_transport\_source\_format (const lbm\_transport\_source\_info\_t \* *info*, size\_t *infosize*, char \* *source*, size\_t \* *size*)**

**Parameters:**

*info* Pointer to a transport source info structure containing the transport source components.  
*infosize* Size (in bytes) of *info*.  
*source* Pointer to a buffer to receive the formatted transport source string.  
*size* Size of *source* in bytes.

**Returns:**

0 for success, -1 for failure.

**8.1.4.267 LBMEExpDLL int lbm\_transport\_source\_parse (const char \* *source*, lbm\_transport\_source\_info\_t \* *info*, size\_t *infosize*)**

**Parameters:**

*source* Source string.  
*info* Pointer to a transport source info structure into which the components are parsed.  
*infosize* Size (in bytes) of *info*.

**Returns:**

0 for success, -1 for failure.

**8.1.4.268 LBMEExpDLL int lbm\_ume\_ack\_delete (lbm\_ume\_rev\_ack\_t \* *ack*)**

**See also:**

[lbm\\_msg\\_extract\\_ume\\_ack](#).

**Parameters:**

*ack* Pointer to the ack structure to be deleted.

**Returns:**

0 for Success, -1 for failure.

**8.1.4.269 LBMEExpDLL int lbm\_ume\_ack\_send\_explicit\_ack  
(lbm\_ume\_rcv\_ack\_t \* ack, lbm\_uint\_t sqn)**

**See also:**

[lbm\\_msg\\_extract\\_ume\\_ack](#).

**Parameters:**

**ack** Pointer to the previously extracted ack structure of a message on the same stream that is currently being explicitly acked.  
**sqn** The sequence number up to which to send the explicit ack.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.270 LBMEExpDLL int lbm\_ume\_src\_msg\_stable (lbm\_src\_t \* src,  
lbm\_uint32\_t sqn)**

**Parameters:**

**src** Pointer to the source.  
**sqn** Sqn of the fragment to mark stable.

**Returns:**

0 for Success, -1 for failure if sqn is not found or sqn is already stable

**8.1.4.271 LBMEExpDLL int lbm\_umq\_ctx\_msg\_stable (lbm\_context\_t \* ctx,  
const char \* qname, lbm\_umqmsgid\_t \* msg\_id)**

**Parameters:**

**ctx** Pointer to the context.  
**qname** Name of the queue.  
**msg\_id** Msg\_id of the message to mark stable.

**Returns:**

0 for Success, -1 for failure if msg\_id is not found or msg\_id is already stable

**8.1.4.272 LBMEExpDLL int lbm\_umq\_msg\_selector\_create  
(lbm\_umq\_msg\_selector\_t \*\* selector, char \* str, lbm\_uint16\_t len)****Parameters:**

*selector* Pointer to a pointer to an lbm\_umq\_msg\_selector\_t structure to be filled in

*str* Pointer to the character string of the message selector.

*len* Length of the above character string.

**Returns:**

the length of message selector string for Success and negative values for Failure.

**8.1.4.273 LBMEExpDLL int lbm\_umq\_msg\_selector\_delete  
(lbm\_umq\_msg\_selector\_t \* selector)****Parameters:**

*selector* Pointer to lbm\_umq\_msg\_selector\_t object.

**8.1.4.274 LBMEExpDLL int lbm\_unicast\_immediate\_message (lbm\_context\_t \*  
ctx, const char \* target, const char \* topic, const char \* data, size\_t len,  
int flags)****Parameters:**

*ctx* Pointer to UM context to send from

*target* Target address of the receiver of the form "TCP:ip:port"

*topic* Topic name to send message to or NULL for non-topic. Topic names should be limited to 246 characters (not including the final null).

*data* Pointer to the data to send in this message

*len* Length (in bytes) of the data to send in this message. Unicast immediate messages must be 65281 bytes or less in length.

*flags* Flags indicating various conditions. ORed set of values.

- LBM\_SRC\_NONBLOCK - If message could not be sent immediately return and error and signal LBM\_EWOULDBLOCK.
- LBM\_SRC\_BLOCK - Block the caller indefinitely until the message is sent. (This behavior is the default if neither LBM\_SRC\_NONBLOCK nor LBM\_SRC\_BLOCK are supplied.)

**Returns:**

-1 for Failure or 0 for Success.

**8.1.4.275 LBMEExpDLL int lbt\_unicast\_immediate\_request (lbt\_request\_t \*\*  
req, lbt\_context\_t \* ctx, const char \* target, const char \* topic, const  
char \* data, size\_t len, lbt\_request\_cb\_proc proc, void \* clientd,  
lbt\_event\_queue\_t \* evq, int flags)**

**Parameters:**

**req** A pointer to a pointer for the lbt\_request\_t object created to be stored.  
**ctx** Pointer to UM context to send from.  
**target** Target address of the receiver of the form "TCP:ip:port"  
**topic** Topic name to send message to or NULL for non-topic. Topic names should  
be limited to 246 characters (not including the final null).  
**data** Buffer to be included as data in the request.  
**len** Length (in bytes) of the data to send in this message. Unicast immediate  
messages must be 65281 bytes or less in length.  
**proc** Pointer to function to call when responses come in for this request.  
**clientd** Client data returned in the callback proc *proc*.  
**evq** optional Event Queue to place message events on when they occur. If NULL  
causes *proc* to be called from context thread.  
**flags** Flags used to instruct UM how to handle this message. See *lbt\_unicast\_-  
immediate\_message* for more information.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.276 LBMEExpDLL const char\* lbt\_version (void)**

**Returns:**

String containing version information for UM.

**8.1.4.277 LBMEExpDLL int lbt\_wildcard\_rcv\_attr\_create  
(lbt\_wildcard\_rcv\_attr\_t \*\* attr)**

The attribute object is allocated and filled with the current default values that are used  
by lbt\_wildcard\_rcv\_t objects and may have been modified by a previously loaded  
configuration file.

**Parameters:**

**attr** A pointer to a pointer to a UM wildcard receiver attribute structure. Will be  
filled in by this function to point to the newly created lbt\_wildcard\_rcv\_-  
attr\_t object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.278 LBMExpDLL int lbm\_wildcard\_rcv\_attr\_create\_default  
(lbm\_wildcard\_rcv\_attr\_t \*\* attr)**

The attribute object is allocated and filled with the initial or factory default values built into LBM that are used by lbm\_wildcard\_rcv\_t.

**Parameters:**

*attr* A pointer to a pointer to a UM wildcard receiver attribute structure. Will be filled in by this function to point to the newly created lbm\_wildcard\_rcv\_attr\_t object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.279 LBMExpDLL int lbm\_wildcard\_rcv\_attr\_create\_from\_xml  
(lbm\_wildcard\_rcv\_attr\_t \*\* attr, const char \* context\_name, const  
char \* pattern, int pattern\_type)**

The attribute object is allocated and filled with the current default values that are used by lbm\_topic\_t objects that concern receivers and may have been modified by a previously loaded configuration file. If an XML configuration file has been loaded, the attribute object is further filled with the defaults for the given context name and wildcard receiver pattern. If the context name or wildcard receiver pattern are not permitted by the XML configuration, -1 is returned and no attribute object is created.

**Parameters:**

*attr* A pointer to a pointer to a UM wildcard receiver attribute structure. Will be filled in by this function to point to the newly created lbm\_wildcard\_rcv\_attr\_t object.

*context\_name* The context name used to lookup the wildcard receiver pattern in the XML configuration. A NULL value is permitted, and will match unnamed contexts defined in the XML.

*pattern* The pattern used to lookup the wildcard receiver in the XML configuration. A NULL value is \*not\* permitted and will result in an error.

*pattern\_type* The type of pattern. Both pattern\_type and pattern must match in XML configuration.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.280 LBMEExpDLL int lbt\_wildcard\_rcv\_attr\_delete  
(lbt\_wildcard\_rcv\_attr\_t \* attr)**

The attribute object is cleaned up and deleted.

**Parameters:**

*attr* Pointer to a UM wildcard receiver attribute object as returned by [lbt\\_wildcard\\_rcv\\_attr\\_create](#).

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.281 LBMEExpDLL int lbt\_wildcard\_rcv\_attr\_dump  
(lbt\_wildcard\_rcv\_attr\_t \* wattr, int \* size, lbt\_config\_option\_t \*  
opts)**

The config object is filled with wildcard receiver configuration options

**Parameters:**

*wattr* The wildcard receiver attribute object to retrieve the attributes from

*size* Size of the opts array. Will return the number of items that were set in opts

*opts* The options array to fill

**8.1.4.282 LBMEExpDLL int lbt\_wildcard\_rcv\_attr\_dup  
(lbt\_wildcard\_rcv\_attr\_t \*\* attr, const lbt\_wildcard\_rcv\_attr\_t \*  
original)**

A new attribute object is created as a copy of an existing object.

**Parameters:**

*attr* A pointer to a pointer to a UM wildcard receiver attribute structure. Will be filled in by this function to point to the newly created lbt\_wildcard\_rcv\_attr\_t object.

*original* Pointer to a UM wildcard receiver attribute object as returned by [lbt\\_wildcard\\_rcv\\_attr\\_create](#) or [lbt\\_wildcard\\_rcv\\_attr\\_create\\_default](#), from which *attr* is initialized.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.283 LBMEExpDLL int lbm\_wildcard\_rcv\_attr\_getopt  
 (lbm\_wildcard\_rcv\_attr\_t \*attr, const char \*optname, void \*optval,  
 size\_t \*optlen)**

**Parameters:**

*attr* Pointer to a UM wildcard receiver attribute object where the option is stored

*optname* String containing the option name

*optval* Pointer to the option value structure. The structure of the option values are specific to the options themselves.

*optlen* When passed, this is the max length (in bytes) of the *optval* structure.  
 When returned, this is the length of the filled in *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.284 LBMEExpDLL int lbm\_wildcard\_rcv\_attr\_option\_size ()**

The function returns the number of entries that are of type "wildcard receiver"

**Returns:**

The number of entries that are of type "wildcard receiver"

**8.1.4.285 LBMEExpDLL int lbm\_wildcard\_rcv\_attr\_set\_from\_xml  
 (lbm\_wildcard\_rcv\_attr\_t \*attr, const char \*context\_name, const  
 char \*pattern, int pattern\_type)**

The attribute object is filled with the defaults for the given context name, wildcard pattern, and pattern\_type, if an XML configuration file has been loaded. If the context name or pattern and pattern\_type combination are not permitted by the XML configuration, -1 is returned and the attribute object is not written to.

**Parameters:**

*attr* A pointer to a UM wildcard receiver attribute structure.

*context\_name* The context name used to lookup the wildcard receiver pattern in the XML configuration. A NULL value is permitted, and will match unnamed contexts defined in the XML.

***pattern*** The pattern used to lookup the wildcard receiver in the XML configuration. A NULL value is \*not\* permitted and will result in an error.

***pattern\_type*** The type of pattern. Both pattern\_type and pattern must match in XML configuration.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.286 LBMEExpDLL int lbt\_wildcard\_rcv\_attr\_setopt**

**(lbt\_wildcard\_rcv\_attr\_t \* attr, const char \* optname, const void \* optval, size\_t optlen)**

Used before the wildcard receiver is created. NOTE: the attribute object must first be initialized with the corresponding \_attr\_create() function.

**Parameters:**

***attr*** Pointer to a UM wildcard receiver attribute object where the option is to be set

***optname*** String containing the option name

***optval*** Pointer to the option value structure. The structure of the option values are specific to the options themselves.

***optlen*** Length (in bytes) of the *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.287 LBMEExpDLL int lbt\_wildcard\_rcv\_attr\_str\_getopt**

**(lbt\_wildcard\_rcv\_attr\_t \* attr, const char \* optname, char \* optval, size\_t \* optlen)**

**Parameters:**

***attr*** Pointer to a UM wildcard receiver attribute object where the option is stored

***optname*** String containing the option name

***optval*** String to be filled in with the option value.

***optlen*** When passed, this is the max length (in bytes) of the string. When returned, this is the length of the filled in option value.

**Returns:**

0 for Success and -1 for Failure.

---

**8.1.4.288 LBMEExpDLL int lbm\_wildcard\_rev\_attr\_str\_setopt  
(lbm\_wildcard\_rev\_attr\_t \* attr, const char \* optname, const char \*  
optval)**

Used before the wildcard receiver is created. NOTE: the attribute object must first be initialized with the corresponding \_attr\_create() function.

**Parameters:**

*attr* Pointer to a UM wildcard receiver attribute object where the option is to be set  
*optname* String containing the option name  
*optval* String containing the option value. The format of the string is specific to the options themselves.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.289 LBMEExpDLL int lbm\_wildcard\_rev\_create (lbm\_wildcard\_rev\_t  
\*\* wrcvp, lbm\_context\_t \* ctx, const char \* pattern, const  
lbm\_rev\_topic\_attr\_t \* tattr, const lbm\_wildcard\_rev\_attr\_t \* wattr,  
lbm\_rev\_cb\_proc proc, void \* clientd, lbm\_event\_queue\_t \* evq)**

The callback *proc* will be called to deliver data sent to the topics that the receiver has requested.

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*wrcvp* A pointer to a pointer to a UM wildcard receiver object. Will be filled in by this function to point to the newly created lbm\_wildcard\_rev\_t object.  
*ctx* Pointer to the LBM context object associated with the wildcard receiver.  
*pattern* Pattern to match the topic strings on for this wildcard. This is by default a regular expression. But more options may be supported.  
*tattr* Pointer to a UM receive topic attribute structure used for specifying attributes for topics created for this wildcard receiver.  
*wattr* Pointer to a UM wildcard receiver attribute structure specifying the options for this wildcard receiver.  
*proc* Pointer to a function to call when messages arrive.

***clientd*** Pointer to client data that is passed when data arrives and *proc* is called.

***evq*** Optional Event Queue to place message events on when they arrive. If NULL causes *proc* to be called from context thread.

**Returns:**

0 for Success and -1 for Failure.

#### 8.1.4.290 **LBMEExpDLL int lbt\_wildcard\_rcv\_delete (lbt\_wildcard\_rcv\_t \* wrcv)**

Delete a UM wildcard receiver object. Note that this function can return while the receiver callback may still be executing if receiver events are being delivered via an event queue. If the application needs to know when all possible processing on the receiver is complete, it must use [lbt\\_wildcard\\_rcv\\_delete\\_ex\(\)](#).

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

***wrcv*** Pointer to a UM wildcard receiver object to delete.

**Returns:**

0 for Success and -1 for Failure.

#### 8.1.4.291 **LBMEExpDLL int lbt\_wildcard\_rcv\_delete\_ex (lbt\_wildcard\_rcv\_t \* wrcv, lbt\_event\_queue\_cancel\_cb\_info\_t \* cbinfo)**

Delete a UM wildcard receiver object, with an application callback indicating when the receiver is fully canceled. This extended version of the receiver delete function requires the configuration option `queue_cancellation_callbacks_enabled` be set to 1.

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

***wrcv*** Pointer to a UM wildcard receiver object to delete.

***cbinfo*** Cancellation callback information, containing the event queue, function pointer, and client data for the callback.

**Returns:**

0 for Success and -1 for Failure.

---

**8.1.4.292 LBMEExpDLL int lbm\_wildcard\_recv\_dump (lbm\_wildcard\_recv\_t \*  
wrcv, int \* size, lbm\_config\_option\_t \* opts)**

The config object is filled with wildcard receiver configuration options

**Parameters:**

*wrcv* The wildcard receiver object to retrieve the attributes from  
*size* Size of the opts array. Will return the number of items that were set in opts  
*opts* The options array to fill

**8.1.4.293 LBMEExpDLL int lbm\_wildcard\_recv\_getopt (lbm\_wildcard\_recv\_t \*  
wrcv, const char \* optname, void \* optval, size\_t \* optlen)**

**Parameters:**

*wrcv* Pointer to a UM wildcard receiver object where the option is stored.  
*optname* String containing the option name.  
*optval* Pointer to the option value structure. The structure of the option values are specific to the options themselves.  
*optlen* When passed, this is the max length (in bytes) of the *optval* structure.  
When returned, this is the length of the filled in *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.294 LBMEExpDLL int lbm\_wildcard\_recv\_setopt (lbm\_wildcard\_recv\_t \*  
wrcv, const char \* optname, const void \* optval, size\_t optlen)**

Only those options that can be set during operation may be specified. See "Options That May Be Set During Operation" (doc/Config/maybesetduringoperation.html) in The UM Configuration Guide. For API functions that can access any option, see *lbm\_wildcard\_recv\_attr\_\**().

**Parameters:**

*wrcv* Pointer to a UM wildcard receiver object where the option is to be set.  
*optname* String containing the option name.  
*optval* Pointer to the option value structure. The structure of the option values are specific to the options themselves.  
*optlen* Length (in bytes) of the *optval* structure.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.295 LBMEExpDLL int lbm\_wildcard\_rcv\_str\_getopt (lbm\_wildcard\_rev\_t \* wrcv, const char \* optname, char \* optval, size\_t \* optlen)**

**Parameters:**

*wrcv* Pointer to a UM wildcard receiver object where the option is stored.

*optname* String containing the option name.

*optval* String to hold the option value.

*optlen* When passed, this is the max length (in bytes) of the string. When returned, this is the length of the filled in option value.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.296 LBMEExpDLL int lbm\_wildcard\_rcv\_str\_setopt (lbm\_wildcard\_rev\_t \* wrcv, const char \* optname, const char \* optval)**

Only those options that can be set during operation may be specified. See "Options That May Be Set During Operation" (doc/Config/maybesetduringoperation.html) in The UM Configuration Guide. For API functions that can access any option, see *lbt\_wildcard\_rcv\_attr\_\**().

**Parameters:**

*wrcv* Pointer to a UM wildcard receiver object where the option is to be set.

*optname* String containing the option name.

*optval* String containing the option value. The format of the string is specific to the options themselves.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.297 LBMEExpDLL int lbm\_wildcard\_rcv\_subscribe\_channel (lbm\_wildcard\_rev\_t \* wrcv, lbm\_uint32\_t channel, lbm\_rcv\_cb\_proc proc, void \* clientd)**

The callback *proc* will be called to deliver messages sent with the specified *channel* number. If NULL is specified for the *proc*, messages with the specified *channel* number will be delivered to the receiver's normal callback. If NULL is specified for the *proc*, any argument passed in for *clientd* will be ignored.

**Warning:**

It is not safe to call this function from a context thread callback.

**Parameters:**

*wrcv* A pointer to a UM wildcard receiver object.

*channel* A channel number to subscribe to.

*proc* Pointer to a function to call when messages arrive.

*clientd* Pointer to clientd data that is passed when data arrives and *proc* is called.

**Returns:**

0 for Success and -1 for Failure

**8.1.4.298 LBMExpDLL int lbm\_wildcard\_rcv\_umq\_deregister  
(lbm\_wildcard\_rcv\_t \* *wrcv*, const char \* *queue\_name*)****Parameters:**

*wrcv* Pointer to wildcard receiver object to de-register.

*queue\_name* Name of the queue to deregister from. A NULL means de-register from all UMQ queues.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.299 LBMExpDLL int lbm\_wildcard\_rcv\_umq\_index\_release  
(lbm\_wildcard\_rcv\_t \* *wrcv*, const char \* *queue\_name*,  
lbm\_umq\_index\_info\_t \* *index\_info*)**

This function causes the UMQ indices to be assigned to another receiver.

**Parameters:**

*wrcv* Pointer to wildcard receiver object that wishes to release the index.

*queue\_name* Name of the queue to reassign the index. A NULL means reassign the index for all UMQ queues.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.300 LBMEExpDLL int lbm\_wildcard\_rcv\_umq\_index\_start\_assignment  
(lbm\_wildcard\_rcv\_t \* wrcv, const char \* queue\_name)****Parameters:**

*wrcv* Pointer to wildcard receiver object to start assignment for.

*queue\_name* Name of the queue to start assignment from. A NULL means start assignment from all UMQ queues.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.301 LBMEExpDLL int lbm\_wildcard\_rcv\_umq\_index\_stop\_assignment  
(lbm\_wildcard\_rcv\_t \* wrcv, const char \* queue\_name)**

This function causes new UMQ indices to not be assigned to the given wildcard receiver from the given UMQ queue(s). Messages with previously assigned UMQ indices may continue to be delivered to the given wildcard receiver from the given UMQ queue(s).

**Parameters:**

*wrcv* Pointer to wildcard receiver object to stop assignment for.

*queue\_name* Name of the queue to stop assignment from. A NULL means stop assignment from all UMQ queues.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.302 LBMEExpDLL int lbm\_wildcard\_rcv\_unsubscribe\_channel  
(lbm\_wildcard\_rcv\_t \* wrcv, lbm\_uint32\_t channel)**

Remove a subscription to a channel previously subscribed to with

**See also:**

[lbm\\_wildcard\\_rcv\\_subscribe\\_channel](#).

**Parameters:**

*wrcv* A pointer to a UM wildcard receiver object.

*channel* The channel number for the channel subscription to be removed.

**Returns:**

0 for Success and -1 for Failure

**8.1.4.303 LBMEExpDLL int lbm\_wildcard\_rcv\_unsubscribe\_channel\_ex  
(lbm\_wildcard\_rcv\_t \* *wrcv*, lbm\_uint32\_t *channel*,  
lbm\_event\_queue\_cancel\_cb\_info\_t \* *cinfo*)**

**Parameters:**

*wrcv* A pointer to a UM wildcard receiver object.

*channel* The channel number for the channel subscription to be removed.

*cinfo* Cancellation callback information, containing the event queue, function pointer, and client data for the callback.

**Returns:**

0 for Success and -1 for Failure

**8.1.4.304 LBMEExpDLL int lbm\_win32\_static\_thread\_attach (void)**

This function sets up UM Thread Local Storage used for handling error information on a per thread basis. This function only needs to be called when using the static version of the UM library on Windows.

**Returns:**

0 for Success and -1 for Failure

**8.1.4.305 LBMEExpDLL int lbm\_win32\_static\_thread\_detach (void)**

This function frees up UM Thread Local Storage used for handling error information on a per thread basis. This function only needs to be called when using the static version of the UM library on Windows.

**Returns:**

0 for Success and -1 for Failure

**8.1.4.306 LBMEExpDLL int lbm\_wrcv\_ume\_deregister (lbm\_wildcard\_rcv\_t \*  
*wrcv*)**

This function causes a UMP deregistration request to be sent to all stores the wildcard receiver is currently registered to, and disallows any future registrations.

**Parameters:**

*wrcv* Pointer to a UM wildcard receiver object

**Returns:**

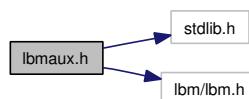
0 for Success and -1 for Failure

## 8.2 lbmaux.h File Reference

Ultra Messaging (UM) Auxiliary Functions API.

```
#include <stdlib.h>
#include <lbm/lbm.h>
```

Include dependency graph for lbmaux.h:



### Functions

- LBMEExpDLL int [lbmaux\\_context\\_create\\_from\\_file](#) (lrbm\_context\_t \*\*Context, const char \*ConfigFile)
 

*Create and initialize an lrbm\_context\_t object, initialized with configuration options from a file.*
- LBMEExpDLL int [lbmaux\\_context\\_attr\\_setopt\\_from\\_file](#) (lrbm\_context\_attr\_t \*Attributes, const char \*ConfigFile)
 

*Set attributes values in an lrbm\_context\_attr\_t object from a configuration file.*
- LBMEExpDLL int [lbmaux\\_src\\_topic\\_attr\\_setopt\\_from\\_file](#) (lrbm\_src\_topic\_attr\_t \*Attributes, const char \*ConfigFile)
 

*Set attributes values in an lrbm\_src\_topic\_attr\_t object from a configuration file.*
- LBMEExpDLL int [lbmaux\\_recv\\_topic\\_attr\\_setopt\\_from\\_file](#) (lrbm\_recv\_topic\_attr\_t \*Attributes, const char \*ConfigFile)
 

*Set attributes values in an lrbm\_recv\_topic\_attr\_t object from a configuration file.*
- LBMEExpDLL int [lbmaux\\_wildcard\\_recv\\_attr\\_setopt\\_from\\_file](#) (lrbm\_wildcard\_recv\_attr\_t \*Attributes, const char \*ConfigFile)
 

*Set attributes values in an lrbm\_wildcard\_recv\_attr\_t object from a configuration file.*
- LBMEExpDLL int [lbmaux\\_event\\_queue\\_attr\\_setopt\\_from\\_file](#) (lrbm\_event\_queue\_attr\_t \*Attributes, const char \*ConfigFile)
 

*Set attributes values in an lrbm\_event\_queue\_attr\_t object from a configuration file.*

### 8.2.1 Detailed Description

**Author:**

David K. Ameiss - Informatica Corporation

**VersIdn:**

//UMprod/REL\_5\_3\_6/29West/lbm/src/auxx/lbm/lbmaux.h#2

The Ultra Messaging (UM) Auxiliary Functions API Description. Included are types, constants, and functions related to the API. Contents are subject to change.

All of the documentation and software included in this and any other Informatica Corporation Ultra Messaging Releases Copyright (C) Informatica Corporation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted only as covered by the terms of a valid software license agreement with Informatica Corporation.

Copyright (C) 2006-2014, Informatica Corporation. All Rights Reserved.

THE SOFTWARE IS PROVIDED "AS IS" AND INFORMATICA DISCLAIMS ALL WARRANTIES EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. INFORMATICA DOES NOT WARRANT THAT USE OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE. INFORMATICA SHALL NOT, UNDER ANY CIRCUMSTANCES, BE LIABLE TO LICENSEE FOR LOST PROFITS, CONSEQUENTIAL, INCIDENTAL, SPECIAL OR INDIRECT DAMAGES ARISING OUT OF OR RELATED TO THIS AGREEMENT OR THE TRANSACTIONS CONTEMPLATED HEREUNDER, EVEN IF INFORMATICA HAS BEEN APPRISED OF THE LIKELIHOOD OF SUCH DAMAGES.

### 8.2.2 Function Documentation

#### 8.2.2.1 LBMEExpDLL int lbmaux\_context\_attr\_setopt\_from\_file (lbm\_context\_attr\_t \*Attributes, const char \*ConfigFile)

This function parses a configuration file, and applies context-scope option values to an lbm\_context\_attr\_t object.

**Parameters:**

**Attributes** A pointer to an initialized lbm\_context\_attr\_t object.

**ConfigFile** String containing the filename that contains the options to parse and set.

**Returns:**

0 for Success and -1 for Failure.

**8.2.2.2 LBMExpDLL int lbmaux\_context\_create\_from\_file (lbum\_context\_t \*\*  
Context, const char \* ConfigFile)**

This function parses a configuration file, and creates an lbum\_context\_attr\_t object with context-scope option values from the configuration file. It then calls [lbum\\_context\\_create\(\)](#) with the attributes object.

**See also:**

[lbum\\_context\\_create\(\)](#)  
[lbum\\_context\\_delete\(\)](#)

**Parameters:**

**Context** A pointer to a pointer to an LBM context object. Will be filled in by this function to point to the newly created lbum\_context\_t object.

**ConfigFile** String containing the filename that contains the options to parse and set.

**Returns:**

0 for Success and -1 for Failure.

**8.2.2.3 LBMExpDLL int lbmaux\_event\_queue\_attr\_setopt\_from\_file  
(lbum\_event\_queue\_attr\_t \* Attributes, const char \* ConfigFile)**

This function parses a configuration file, and applies event-queue-scope option values to an lbum\_event\_queue\_attr\_t object.

**Parameters:**

**Attributes** A pointer to an initialized lbum\_event\_queue\_attr\_t object.

**ConfigFile** String containing the filename that contains the options to parse and set.

**Returns:**

0 for Success and -1 for Failure.

**8.2.2.4 LBMEExpDLL int lbmaux\_recv\_topic\_attr\_setopt\_from\_file  
(lbtm\_recv\_topic\_attr\_t \*Attributes, const char \*ConfigFile)**

This function parses a configuration file, and applies receiver-scope option values to an lbtm\_recv\_topic\_attr\_t object.

**Parameters:**

*Attributes* A pointer to an initialized lbtm\_recv\_topic\_attr\_t object.

*ConfigFile* String containing the filename that contains the options to parse and set.

**Returns:**

0 for Success and -1 for Failure.

**8.2.2.5 LBMEExpDLL int lbmaux\_src\_topic\_attr\_setopt\_from\_file  
(lbtm\_src\_topic\_attr\_t \*Attributes, const char \*ConfigFile)**

This function parses a configuration file, and applies source-scope option values to an lbtm\_src\_topic\_attr\_t object.

**Parameters:**

*Attributes* A pointer to an initialized lbtm\_src\_topic\_attr\_t object.

*ConfigFile* String containing the filename that contains the options to parse and set.

**Returns:**

0 for Success and -1 for Failure.

**8.2.2.6 LBMEExpDLL int lbmaux\_wildcard\_recv\_attr\_setopt\_from\_file  
(lbtm\_wildcard\_recv\_attr\_t \*Attributes, const char \*ConfigFile)**

This function parses a configuration file, and applies wildcard-receiver-scope option values to an lbtm\_wildcard\_recv\_attr\_t object.

**Parameters:**

*Attributes* A pointer to an initialized lbtm\_wildcard\_recv\_attr\_t object.

*ConfigFile* String containing the filename that contains the options to parse and set.

**Returns:**

0 for Success and -1 for Failure.

## 8.3 lbmht.h File Reference

Ultra Messaging (UM) HyperTopic API.

```
#include "lbt/lbt.h"
```

Include dependency graph for lbmht.h:



### Data Structures

- struct `lbt_delete_cb_info_t_stct`

*Structure passed to the `lbt_hypertopic_rcv_delete()` function so that a deletion callback may be called.*

### Defines

- #define `LBT_HT_BASE_MATCH_LEVEL` 0
- #define `LBT_HT_TOKEN_GLOBNAME` 0
- #define `LBT_HT_TOKEN_GLOPATH` 1
- #define `LBT_HT_TOKEN_NAME` 2
- #define `LBT_HT_INIT_BRANCH_SZ` 16
- #define `LBT_HT_CBVEC_SZ` 16
- #define `LBT_HT_CBVEC_FLAG_ACTIVE` 1
- #define `LBT_HT_CBVEC_FLAG_DELETED` 2

### Typedefs

- typedef `lbt_hypertopic_rcv_stct` `lbt_hypertopic_rcv_t`  
*HyperTopic receiver object (opaque).*
- typedef `int(*) lbt_hypertopic_rcv_cb_proc (lbt_hypertopic_rcv_t *hrcv, lbt_msg_t *msg, void *clientd)`  
*Application callback for messages delivered to HyperTopic receivers.*
- typedef `void(*) lbt_delete_cb_proc (int dispatch_thrd, void *clientd)`  
*Application callback for `lbt_hypertopic_rcv_delete()`.*
- typedef `lbt_delete_cb_info_t_stct lbt_delete_cb_info_t`

*Structure passed to the [lbt\\_hypertopic\\_rcv\\_delete\(\)](#) function so that a deletion callback may be called.*

## Functions

- LBMDLL int [lbt\\_hypertopic\\_rcv\\_init](#) (lbt\_hypertopic\_rcv\_t \*\*hrcvp, lbt\_context\_t \*ctx, const char \*prefix, lbt\_event\_queue\_t \*evq)  
*Initialize a HyperTopic receiver.*
- LBMDLL int [lbt\\_hypertopic\\_rcv\\_add](#) (lbt\_hypertopic\_rcv\_t \*hrcv, const char \*pattern, [lbt\\_hypertopic\\_rcv\\_cb\\_proc](#) proc, void \*clientd)  
*Add a topic pattern to the set of topics being received by a HyperTopic receiver.*
- LBMDLL int [lbt\\_hypertopic\\_rcv\\_delete](#) (lbt\_hypertopic\_rcv\_t \*hrcv, const char \*pattern, [lbt\\_hypertopic\\_rcv\\_cb\\_proc](#) proc, void \*clientd, [lbt\\_delete\\_cb\\_info\\_t](#) \*cbinfo)  
*Delete a previously added topic from a HyperTopic receiver topic set.*
- LBMDLL int [lbt\\_hypertopic\\_rcv\\_destroy](#) (lbt\_hypertopic\_rcv\_t \*hrcv)  
*Clean-up HyperTopic receiver previously created by [lbt\\_hypertopic\\_rcv\\_init\(\)](#).*

### 8.3.1 Detailed Description

#### Author:

M. Garwood - Informatica Corporation

#### VersIOn:

//UMprod/REL\_5\_3\_6/29West/lbm/src/lib/lbm/lbmht.h#2

The Ultra Messaging (UM) HyperTopic API Description. Included are types, constants, and functions related to the API. Contents are subject to change.

All of the documentation and software included in this and any other Informatica Corporation Ultra Messaging Releases Copyright (C) Informatica Corporation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted only as covered by the terms of a valid software license agreement with Informatica Corporation.

Copyright (C) 2006-2014, Informatica Corporation. All Rights Reserved.

THE SOFTWARE IS PROVIDED "AS IS" AND INFORMATICA DISCLAIMS ALL WARRANTIES EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. INFORMATICA DOES NOT WARRANT THAT USE OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE. INFORMATICA SHALL NOT, UNDER ANY CIRCUMSTANCES, BE LIABLE TO LICENSEE FOR LOST PROFITS, CONSEQUENTIAL, INCIDENTAL, SPECIAL OR INDIRECT DAMAGES ARISING OUT OF OR RELATED TO THIS AGREEMENT OR THE TRANSACTIONS CONTEMPLATED HEREUNDER, EVEN IF INFORMATICA HAS BEEN APPRISED OF THE LIKELIHOOD OF SUCH DAMAGES.

The LBM HyperTopic API provides a mechanism to enable the efficient reception of messages sent over any LBM transport (especially an immediate messaging transport) where the message topic conforms to a simple hierarchical wildcard topic structure.

The hierarchical topic supported by this API has a BNF grammar as follows:

```

<ht-topic> ::= <ht-prefix> <ht-topic-comps>
             | <ht-topic-comps>
<ht-topic-comps> ::= <ht-topic-comp> "/" <ht-topic-comps>
                     | <ht-topic-comp> "/>"
                     | <ht-topic-comp>
                     | ">"
<ht-topic-comp> ::= <text>
                     | "*"
                     | ""
<ht-prefix> ::= <text>

```

The "\*" token in the above grammar will match any sequence of characters excluding a "/". The ">" token will match any sequence of characters including a "/" (until the end of the topic string).

The grammar supports a prefix string that can be applied to effectively implement a namespace for each instance of a HyperTopic receiver.

A HyperTopic receiver is initialized using the [lbt\\_hypertopic\\_rcv\\_init\(\)](#) API function. This function sets the HyperTopic namespace prefix and an optional event queue to be used for messages received on all topics which are part of the HyperTopic namespace.

The [lbt\\_hypertopic\\_rcv\\_add\(\)](#) and [lbt\\_hypertopic\\_rcv\\_delete\(\)](#) functions are used to add and remove topic patterns to the HyperTopic namespace. Unlike non-HyperTopic receiver creation and deletion functions, these functions take the same set of arguments which include a pointer to the HyperTopic receiver object, the topic pattern, received message callback function and client data pointer (passed with message to the callback function). The [lbt\\_hypertopic\\_rcv\\_delete\(\)](#) API function also accepts an optional designation for a callback function to be called once all message callbacks for the topic being deleted have completed.

The [lbt\\_hypertopic\\_rcv\\_destroy\(\)](#) API function should be called to de-initialize and clean-up the HyperTopic receiver after all topics added to the HyperTopic namespace

have been deleted.

### 8.3.2 Typedef Documentation

#### 8.3.2.1 **typedef void(\*) lhm\_delete\_cb\_proc(int dispatch\_thrd, void \*clientd)**

Set by [lhm\\_hypertopic\\_rcv\\_delete\(\)](#). Note: This application callback can be made from the context thread, and is therefore, limited in the LBM API calls that it can make. The application is called after all events associated with the delete are completed.

**See also:**

[lhm\\_delete\\_cb\\_info\\_t](#)

**Parameters:**

*dispatch\_thrd* Indicates from where the callback is being called. This can be useful to the application to avoid deadlock.

- 1 - Called by the dispatch thread (after the call to [lhm\\_hypertopic\\_rcv\\_delete\(\)](#) returns).
- 0 - Called directly by the [lhm\\_hypertopic\\_rcv\\_delete\(\)](#) function.

*clientd* Client data pointer supplied in the [lhm\\_delete\\_cb\\_info\\_t](#) passed to [lhm\\_hypertopic\\_rcv\\_delete\(\)](#).

**Returns:**

0 always.

#### 8.3.2.2 **typedef int(\*) lhm\_hypertopic\_rcv\_cb\_proc(lhm\_hypertopic\_rcv\_t \*hrcv, lhm\_msg\_t \*msg, void \*clientd)**

Set by [lhm\\_hypertopic\\_rcv\\_add\(\)](#). If this application callback is set on an HyperTopic receiver that has been initialized without an event queue, it is called from the context thread and is therefore, limited in the API calls that it can make.

After the callback returns, the message object *msg* is deleted and the application must not refer to it. This behavior can be overridden by calling [lhm\\_msg\\_retain\(\)](#) from the receive callback before it returns. It then becomes the application's responsibility to delete the message object by calling [lhm\\_msg\\_delete\(\)](#) after the application no longer needs to refer to the message structure or its contents.

**Note:**

For received application messages, be aware that LBM does not guarantee any alignment of that data.

**Parameters:**

*hrcv* HyperTopic receiver object generating the event.  
*msg* Message object containing the receiver event.  
*clientd* Client data pointer supplied in [lbm\\_hypertopic\\_rcv\\_add\(\)](#).

**Returns:**

0 always.

### 8.3.3 Function Documentation

#### 8.3.3.1 LBMExpDLL int [lbm\\_hypertopic\\_rcv\\_add](#) (lbm\_hypertopic\_rcv\_t \* *hrcv*, const char \* *pattern*, lbm\_hypertopic\_rcv\_cb\_proc *proc*, void \* *clientd*)

**Parameters:**

*hrcv* HyperTopic object created by [lbm\\_hypertopic\\_rcv\\_init\(\)](#)  
*pattern* Hierarchical topic pattern to add to the HyperTopic group.  
*proc* Pointer to a function to call when messages arrive on a topic matched by  
*pattern*.  
*clientd* Pointer to client data that is passed to *proc* when data arrives on the topic  
matched by *pattern*.

**Returns:**

0 for success, -1 on failure

#### 8.3.3.2 LBMExpDLL int [lbm\\_hypertopic\\_rcv\\_delete](#) (lbm\_hypertopic\_rcv\_t \* *hrcv*, const char \* *pattern*, lbm\_hypertopic\_rcv\_cb\_proc *proc*, void \* *clientd*, lbm\_delete\_cb\_info\_t \* *cbinfo*)

**Parameters:**

*hrcv* HyperTopic object created by [lbm\\_hypertopic\\_rcv\\_init\(\)](#)  
*pattern* Hierarchical topic pattern to delete from the HyperTopic group.  
*proc* Pointer to a function being called when messages arrive from the given *pat-*  
*tern*.  
*clientd* Pointer to client data that is being passed to *proc* when data arrives on a  
topic matched by *pattern*.

**Returns:**

0 for success, -1 on failure

**8.3.3.3 LBMEExpDLL int lbm\_hypertopic\_rcv\_destroy (lbm\_hypertopic\_rcv\_t \* *hrcv*)****Parameters:**

*hrcv* HyperTopic receiver to be destroyed.

**Returns:**

0 for success, -1 on failure

**8.3.3.4 LBMEExpDLL int lbm\_hypertopic\_rcv\_init (lbm\_hypertopic\_rcv\_t \*\* *hrcvp*, lbm\_context\_t \* *ctx*, const char \* *prefix*, lbm\_event\_queue\_t \* *evq*)****Parameters:**

*hrcvp* Pointer to location where the lbm\_hypertopic\_rcv\_t object will be returned.

*ctx* Pointer to the LBM context object associated with the receiver.

*prefix* Namespace prefix for the HyperTopic receiver. The prefix string constrains the topic namespace to topics that begin with the specified prefix only. This parameter may be set to NULL if no prefix is to be defined for this HyperTopic namespace.

*evq* Optional Event Queue to place message events on when they arrive. If NULL, all messages will be delivered from the context thread.

**Returns:**

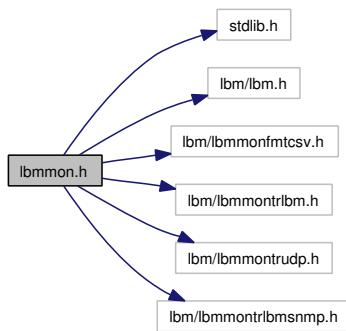
0 for success, -1 on failure

## 8.4 lbmmon.h File Reference

Ultra Messaging (UM) Monitoring API.

```
#include <stdlib.h>
#include <lbm/lbm.h>
#include <lbm/lbmmonfmtcsv.h>
#include <lbm/lbmmontrlrbm.h>
#include <lbm/lbmmontrudp.h>
#include <lbm/lbmmontrlbmsnmp.h>
```

Include dependency graph for lbmmon.h:



## Data Structures

- struct [lbmmon\\_packet\\_hdr\\_t\\_stct](#)  
*Statistics packet header layout.*
- struct [lbmmon\\_attr\\_block\\_t\\_stct](#)  
*Statistics attribute block layout. Associated with each statistics message is a set of optional attributes. Any attributes present will immediately follow the packet header.*
- struct [lbmmon\\_attr\\_entry\\_t\\_stct](#)  
*Statistics attribute entry layout. Each attribute entry within the attributes block consists of an entry header, followed immediately by the attribute data.*
- struct [lbmmon\\_format\\_func\\_t\\_stct](#)  
*Format module function pointer container.*
- struct [lbmmon\\_rcv\\_statistics\\_func\\_t\\_stct](#)

*A structure that holds the callback information for receiver statistics.*

- struct [lbmon\\_src\\_statistics\\_func\\_t\\_stct](#)  
*A structure that holds the callback information for source statistics.*
- struct [lbmon\\_evq\\_statistics\\_func\\_t\\_stct](#)  
*A structure that holds the callback information for event queue statistics.*
- struct [lbmon\\_ctx\\_statistics\\_func\\_t\\_stct](#)  
*A structure that holds the callback information for context statistics.*
- struct [lbmon\\_transport\\_func\\_t\\_stct](#)  
*Transport module function pointer container.*

## Defines

- #define [LBMMON\\_ERROR\\_BASE](#) 4096
- #define [LBMMON\\_EINVAL](#) (LBMMON\_ERROR\_BASE + 1)
- #define [LBMMON\\_ENOMEM](#) (LBMMON\_ERROR\_BASE + 2)
- #define [LBMMON\\_EMODFAIL](#) (LBMMON\_ERROR\_BASE + 3)
- #define [LBMMON\\_ELBMFAIL](#) (LBMMON\_ERROR\_BASE + 4)
- #define [LBMMON\\_EAGAIN](#) (LBMMON\_ERROR\_BASE + 5)
- #define [LBMMON\\_EALREADY](#) (LBMMON\_ERROR\_BASE + 6)
- #define [LBMMON\\_PACKET\\_SIGNATURE](#) 0x1b33041b
- #define [LBMMON\\_PACKET\\_TYPE\\_SOURCE](#) 0
- #define [LBMMON\\_PACKET\\_TYPE\\_RECEIVER](#) 1
- #define [LBMMON\\_PACKET\\_TYPE\\_EVENT\\_QUEUE](#) 2
- #define [LBMMON\\_PACKET\\_TYPE\\_CONTEXT](#) 3
- #define [LBMMON\\_ATTR\\_SENDER\\_IPV4](#) 0
- #define [LBMMON\\_ATTR\\_TIMESTAMP](#) 1
- #define [LBMMON\\_ATTR\\_APPSOURCEID](#) 2
- #define [LBMMON\\_ATTR\\_FORMAT\\_MODULEID](#) 3
- #define [LBMMON\\_ATTR\\_OBJECTID](#) 4
- #define [LBMMON\\_ATTR\\_CONTEXTID](#) LBMMON\_ATTR\_OBJECTID
- #define [LBMMON\\_ATTR\\_PROCESSID](#) 5
- #define [LBMMON\\_ATTR\\_SOURCE](#) 6
- #define [LBMMON\\_ATTR\\_SOURCE\\_NORMAL](#) 0
- #define [LBMMON\\_ATTR\\_SOURCE\\_IM](#) 1
- #define [LBMMON\\_RCTL\\_RECEIVER\\_CALLBACK](#) 0
- #define [LBMMON\\_RCTL\\_SOURCE\\_CALLBACK](#) 1
- #define [LBMMON\\_RCTL\\_EVENT\\_QUEUE\\_CALLBACK](#) 2
- #define [LBMMON\\_RCTL\\_CONTEXT\\_CALLBACK](#) 3

## Typedefs

- **typedef lbmmon\_packet\_hdr\_t\_stct lbmmon\_packet\_hdr\_t**  
*Statistics packet header layout.*
- **typedef lbmmon\_attr\_block\_t\_stct lbmmon\_attr\_block\_t**  
*Statistics attribute block layout. Associated with each statistics message is a set of optional attributes. Any attributes present will immediately follow the packet header.*
- **typedef lbmmon\_attr\_entry\_t\_stct lbmmon\_attr\_entry\_t**  
*Statistics attribute entry layout. Each attribute entry within the attributes block consists of an entry header, followed immediately by the attribute data.*
- **typedef int(\*) lbmmon\_format\_init\_t (void \*\*FormatClientData, const void \*FormatOptions)**  
*Function to initialize a format module.*
- **typedef int(\*) lbmmon\_rcv\_format\_serialize\_t (char \*Destination, size\_t \*Size, unsigned short \*ModuleID, const lbm\_rcv\_transport\_stats\_t \*Statistics, void \*FormatClientData)**  
*Function to serialize an lbm\_rcv\_transport\_stats\_t structure.*
- **typedef int(\*) lbmmon\_src\_format\_serialize\_t (char \*Destination, size\_t \*Size, unsigned short \*ModuleID, const lbm\_src\_transport\_stats\_t \*Statistics, void \*FormatClientData)**  
*Function to serialize an lbm\_src\_transport\_stats\_t structure.*
- **typedef int(\*) lbmmon\_evq\_format\_serialize\_t (char \*Destination, size\_t \*Size, unsigned short \*ModuleID, const lbm\_event\_queue\_stats\_t \*Statistics, void \*FormatClientData)**  
*Function to serialize an lbm\_event\_queue\_stats\_t structure.*
- **typedef int(\*) lbmmon\_ctx\_format\_serialize\_t (char \*Destination, size\_t \*Size, unsigned short \*ModuleID, const lbm\_context\_stats\_t \*Statistics, void \*FormatClientData)**  
*Function to serialize an lbm\_context\_stats\_t structure.*
- **typedef int(\*) lbmmon\_rcv\_format\_deserialize\_t (lbm\_rcv\_transport\_stats\_t \*Statistics, const char \*Source, size\_t Length, unsigned short ModuleID, void \*FormatClientData)**  
*Function to deserialize a buffer into an lbm\_rcv\_transport\_stats\_t structure.*
- **typedef int(\*) lbmmon\_src\_format\_deserialize\_t (lbm\_src\_transport\_stats\_t \*Statistics, const char \*Source, size\_t Length, unsigned short ModuleID, void \*FormatClientData)**

*Function to deserialize a buffer into an `lbt_src_transport_stats_t` structure.*

- `typedef int(*) lbmon_evq_format_deserialize_t (lbt_event_queue_stats_t *Statistics, const char *Source, size_t Length, unsigned short ModuleID, void *FormatClientData)`

*Function to deserialize a buffer into an `lbt_event_queue_stats_t` structure.*

- `typedef int(*) lbmon_ctx_format_deserialize_t (lbt_context_stats_t *Statistics, const char *Source, size_t Length, unsigned short ModuleID, void *FormatClientData)`

*Function to deserialize a buffer into an `lbt_context_stats_t` structure.*

- `typedef int(*) lbmon_format_finish_t (void *FormatClientData)`

*Function to finish format module processing.*

- `typedef const char *(*lbmon_format_errmsg_t) (void)`

*Function to return the last error message from a format module.*

- `typedef lbmon_format_func_t_stct lbmon_format_func_t`

*Format module function pointer container.*

- `typedef void(*) lbmon_rcv_statistics_cb (const void *AttributeBlock, const lbt_rcv_transport_stats_t *Statistics, void *ClientData)`

*Client callback function to process a received receiver statistics packet.*

- `typedef lbmon_rcv_statistics_func_t_stct lbmon_rcv_statistics_func_t`

*A structure that holds the callback information for receiver statistics.*

- `typedef void(*) lbmon_src_statistics_cb (const void *AttributeBlock, const lbt_src_transport_stats_t *Statistics, void *ClientData)`

*Client callback function to process a received source statistics packet.*

- `typedef lbmon_src_statistics_func_t_stct lbmon_src_statistics_func_t`

*A structure that holds the callback information for source statistics.*

- `typedef void(*) lbmon_evq_statistics_cb (const void *AttributeBlock, const lbt_event_queue_stats_t *Statistics, void *ClientData)`

*Client callback function to process a received event queue statistics packet.*

- `typedef lbmon_evq_statistics_func_t_stct lbmon_evq_statistics_func_t`

*A structure that holds the callback information for event queue statistics.*

- **typedef void(\*) lbmmon\_ctx\_statistics\_cb** (const void \*AttributeBlock, const [lbp\\_context\\_stats\\_t](#) \*Statistics, void \*ClientData)

*Client callback function to process a received context statistics packet.*

- **typedef lbmmon\_ctx\_statistics\_func\_t\_stct lbmmon\_ctx\_statistics\_func\_t**

*A structure that holds the callback information for context statistics.*

- **typedef int(\*) lbmmon\_transport\_initsrc\_t** (void \*\*TransportClientData, const void \*TransportOptions)

*Function to initialize a transport module to serve as a source of statistics.*

- **typedef int(\*) lbmmon\_transport\_initrcv\_t** (void \*\*TransportClientData, const void \*TransportOptions)

*Function to initialize a transport module to serve as a receiver of statistics.*

- **typedef int(\*) lbmmon\_transport\_send\_t** (const char \*Data, size\_t Length, void \*TransportClientData)

*Send a statistics packet.*

- **typedef int(\*) lbmmon\_transport\_receive\_t** (char \*Data, size\_t \*Length, unsigned int TimeoutMS, void \*TransportClientData)

*Receive statistics data.*

- **typedef int(\*) lbmmon\_transport\_finishsrc\_t** (void \*TransportClientData)

*Finish processing for a source transport.*

- **typedef int(\*) lbmmon\_transport\_finishrcv\_t** (void \*TransportClientData)

*Finish processing for a receiver transport.*

- **typedef const char \*(\*) lbmmon\_transport\_errmsg\_t** (void)

*Function to return the last error message from a transport module.*

- **typedef lbmmon\_transport\_func\_t\_stct lbmmon\_transport\_func\_t**

*Transport module function pointer container.*

  - **typedef lbmmon\_sctl\_t\_stct lbmmon\_sctl\_t**
  - **typedef lbmmon\_rctl\_attr\_t\_stct lbmmon\_rctl\_attr\_t**
  - **typedef lbmmon\_rctl\_t\_stct lbmmon\_rctl\_t**

## Functions

- LBMDLL int [lbmon\\_sctl\\_create](#) (lbmon\_sctl\_t \*\*Control, const lbmon\_format\_func\_t \*Format, const void \*FormatOptions, const lbmon\_transport\_func\_t \*Transport, const void \*TransportOptions)

*Create an LBM Monitoring Source Controller.*
- LBMDLL int [lbmon\\_rctl\\_attr\\_create](#) (lbmon\_rctl\_attr\_t \*\*Attributes)

*Create an LBM Monitoring Receive Controller attribute object.*
- LBMDLL int [lbmon\\_rctl\\_attr\\_delete](#) (lbmon\_rctl\_attr\_t \*Attributes)

*Delete an LBM Monitoring Receive Controller attribute object.*
- LBMDLL int [lbmon\\_rctl\\_attr\\_setopt](#) (lbmon\_rctl\_attr\_t \*Attributes, int Option, void \*Value, size\_t Length)

*Set an LBMMON receive controller attribute option value.*
- LBMDLL int [lbmon\\_rctl\\_attr\\_getopt](#) (lbmon\_rctl\_attr\_t \*Attributes, int Option, void \*Value, size\_t \*Length)

*Get an LBMMON receive controller attribute option value.*
- LBMDLL int [lbmon\\_rctl\\_create](#) (lbmon\_rctl\_t \*\*Control, const lbmon\_format\_func\_t \*Format, const void \*FormatOptions, const lbmon\_transport\_func\_t \*Transport, const void \*TransportOptions, lbmon\_rctl\_attr\_t \*Attributes, void \*ClientData)

*Create an LBM Monitoring Receive Controller.*
- LBMDLL int [lbmon\\_context\\_monitor](#) (lbmon\_sctl\_t \*Control, lbm\_context\_t \*Context, const char \*ApplicationSourceID, unsigned int Seconds)

*Register a context for monitoring.*
- LBMDLL int [lbmon\\_context\\_unmonitor](#) (lbmon\_sctl\_t \*Control, lbm\_context\_t \*Context)

*Terminate monitoring for a context.*
- LBMDLL int [lbmon\\_src\\_monitor](#) (lbmon\_sctl\_t \*Control, lbm\_src\_t \*Source, const char \*ApplicationSourceID, unsigned int Seconds)

*Register a source for monitoring.*
- LBMDLL int [lbmon\\_src\\_unmonitor](#) (lbmon\_sctl\_t \*Control, lbm\_src\_t \*Source)

*Terminate monitoring for a source.*

- LBMEExpDLL int [lbmon\\_rev\\_monitor](#) (lbmon\_sctl\_t \*Control, lbm\_rcv\_t \*Receiver, const char \*ApplicationSourceID, unsigned int Seconds)  
*Register a receiver for monitoring.*
- LBMEExpDLL int [lbmon\\_rcv\\_unmonitor](#) (lbmon\_sctl\_t \*Control, lbm\_rcv\_t \*Receiver)  
*Terminate monitoring for a receiver.*
- LBMEExpDLL int [lbmon\\_evq\\_monitor](#) (lbmon\_sctl\_t \*Control, lbm\_event\_queue\_t \*EventQueue, const char \*ApplicationSourceID, unsigned int Seconds)  
*Register an event queue for monitoring.*
- LBMEExpDLL int [lbmon\\_evq\\_unmonitor](#) (lbmon\_sctl\_t \*Control, lbm\_event\_queue\_t \*EventQueue)  
*Terminate monitoring for an event queue.*
- LBMEExpDLL int [lbmon\\_sctl\\_destroy](#) (lbmon\_sctl\_t \*Control)  
*Destroy a source monitoring controller.*
- LBMEExpDLL int [lbmon\\_rctl\\_destroy](#) (lbmon\_rctl\_t \*Control)  
*Destroy a statistics receive controller.*
- LBMEExpDLL int [lbmon\\_sctl\\_sample](#) (lbmon\_sctl\_t \*Control)  
*Gather statistics for on-demand objects.*
- LBMEExpDLL int [lbmon\\_sctl\\_sample\\_ex](#) (lbmon\_sctl\_t \*Control, const char \*ApplicationSourceID)  
*Extended gather statistics for on-demand objects.*
- LBMEExpDLL int [lbmon\\_attr\\_get\\_ip4sender](#) (const void \*AttributeBlock, lbm\_uint\_t \*Address)  
*Retrieve the IPV4 sender address attribute from the statistics attribute block.*
- LBMEExpDLL int [lbmon\\_attr\\_get\\_timestamp](#) (const void \*AttributeBlock, time\_t \*Timestamp)  
*Retrieve the timestamp attribute from the statistics attribute block.*
- LBMEExpDLL int [lbmon\\_attr\\_get\\_appsourceid](#) (const void \*AttributeBlock, char \*ApplicationSourceID, size\_t Length)  
*Retrieve the application source ID attribute from the statistics attribute block.*

- LBMDLL int **lbmon\_attr\_get\_objectid** (const void \*AttributeBlock, lbm\_ulong\_t \*ObjectID)

*Retrieve the object ID attribute from the statistics attribute block.*
- LBMDLL int **lbmon\_attr\_get\_contextid** (const void \*AttributeBlock, lbm\_ulong\_t \*ContextID)

*Retrieve the context ID attribute from the statistics attribute block.*
- LBMDLL int **lbmon\_attr\_get\_processid** (const void \*AttributeBlock, lbm\_ulong\_t \*ProcessID)

*Retrieve the process ID attribute from the statistics attribute block.*
- LBMDLL int **lbmon\_attr\_get\_source** (const void \*AttributeBlock, lbm\_ulong\_t \*Source)

*Retrieve the source attribute from the statistics attribute block.*
- LBMDLL int **lbmon\_errnum** (void)

*Retrieve the error number for the last error encountered.*
- LBMDLL const char \* **lbmon\_errmsg** (void)

*Retrieve the error message for the last error encountered.*
- const char \* **lbmon\_next\_key\_value\_pair** (const char \*String, char \*Key, size\_t KeySize, char \*Value, size\_t ValueSize)

*Retrieve the next key/value pair from a semicolon-separated list.*

### 8.4.1 Detailed Description

#### Author:

David K. Ameiss - Informatica Corporation

#### VersIdn:

//UMprod/REL\_5\_3\_6/29West/lbm/src/mon/lbm/lbmon.h#2

The Ultra Messaging (UM) Monitoring API Description. Included are types, constants, and functions related to the API. Contents are subject to change.

All of the documentation and software included in this and any other Informatica Corporation Ultra Messaging Releases Copyright (C) Informatica Corporation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted only as covered by the terms of a valid software license agreement with Informatica Corporation.

Copyright (C) 2006-2014, Informatica Corporation. All Rights Reserved.

THE SOFTWARE IS PROVIDED "AS IS" AND INFORMATICA DISCLAIMS ALL WARRANTIES EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. INFORMATICA DOES NOT WARRANT THAT USE OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE. INFORMATICA SHALL NOT, UNDER ANY CIRCUMSTANCES, BE LIABLE TO LICENSEE FOR LOST PROFITS, CONSEQUENTIAL, INCIDENTAL, SPECIAL OR INDIRECT DAMAGES ARISING OUT OF OR RELATED TO THIS AGREEMENT OR THE TRANSACTIONS CONTEMPLATED HEREUNDER, EVEN IF INFORMATICA HAS BEEN APPRISED OF THE LIKELIHOOD OF SUCH DAMAGES.

The LBM Monitoring API provides a framework to allow the convenient gathering of LBM statistics.

### 8.4.2 Define Documentation

#### 8.4.2.1 #define LBMMON\_ATTR\_APPSOURCEID 2

Attribute block entry contains the application source ID string.

#### 8.4.2.2 #define LBMMON\_ATTR\_CONTEXTID LBMMON\_ATTR\_OBJECTID

Attribute block entry contains the context ID.

##### Deprecated

Use [LBMMON\\_ATTR\\_OBJECTID](#) instead.

#### 8.4.2.3 #define LBMMON\_ATTR\_FORMAT\_MODULEID 3

Attribute block entry contains the format module ID.

#### 8.4.2.4 #define LBMMON\_ATTR\_OBJECTID 4

Attribute block contains the object ID.

**8.4.2.5 #define LBMMON\_ATTR\_PROCESSID 5**

Attribute block entry contains the process ID.

**8.4.2.6 #define LBMMON\_ATTR\_SENDER\_IPV4 0**

Attribute block entry contains the sender IPV4 address.

**8.4.2.7 #define LBMMON\_ATTR\_SOURCE 6**

Attribute block entry contains the source flag. See LBMMON\_ATTR\_SOURCE\_\* for possible values. Used to distinguish between MIM source/receiver statistics and normal transport source/receiver statistics.

**8.4.2.8 #define LBMMON\_ATTR\_SOURCE\_IM 1**

Source/receiver statistics are from a MIM transport session.

**8.4.2.9 #define LBMMON\_ATTR\_SOURCE\_NORMAL 0**

Source/receiver statistics are from a normal transport session.

**8.4.2.10 #define LBMMON\_ATTR\_TIMESTAMP 1**

Attribute block entry contains the timestamp.

**8.4.2.11 #define LBMMON\_EAGAIN (LBMMON\_ERROR\_BASE + 5)**

[lbmon\\_errnum\(\)](#) value. Insufficient resources.

**8.4.2.12 #define LBMMON\_EALREADY (LBMMON\_ERROR\_BASE + 6)**

[lbmon\\_errnum\(\)](#) value. Resource already registered.

**8.4.2.13 #define LBMMON\_EINVAL (LBMMON\_ERROR\_BASE + 1)**

[lbmon\\_errnum\(\)](#) value. An invalid argument was passed.

**8.4.2.14 #define LBMMON\_ELBMFAIL (LBMMON\_ERROR\_BASE + 4)**

[lbmmmon\\_errnum\(\)](#) value. A call to an LBM function failed.

**8.4.2.15 #define LBMMON\_EMODFAIL (LBMMON\_ERROR\_BASE + 3)**

[lbmmmon\\_errnum\(\)](#) value. A call to a module function failed.

**8.4.2.16 #define LBMMON\_ENOMEM (LBMMON\_ERROR\_BASE + 2)**

[lbmmmon\\_errnum\(\)](#) value. Out of memory.

**8.4.2.17 #define LBMMON\_ERROR\_BASE 4096**

Base value for LBMMON error codes.

**8.4.2.18 #define LBMMON\_PACKET\_SIGNATURE 0x1b33041b**

Packet signature value

**8.4.2.19 #define LBMMON\_PACKET\_TYPE\_CONTEXT 3**

Packet contains context statistics

**8.4.2.20 #define LBMMON\_PACKET\_TYPE\_EVENT\_QUEUE 2**

Packet contains event queue statistics

**8.4.2.21 #define LBMMON\_PACKET\_TYPE\_RECEIVER 1**

Packet contains receiver statistics

**8.4.2.22 #define LBMMON\_PACKET\_TYPE\_SOURCE 0**

Packet contains source statistics

**8.4.2.23 #define LBMMON\_RCTL\_CONTEXT\_CALLBACK 3**

Receive controller attribute option: Context statistics callback.

**8.4.2.24 #define LBMMON\_RCTL\_EVENT\_QUEUE\_CALLBACK 2**

Receive controller attribute option: Event queue statistics callback.

**8.4.2.25 #define LBMMON\_RCTL\_RECEIVER\_CALLBACK 0**

Receive controller attribute option: Receiver statistics callback.

**8.4.2.26 #define LBMMON\_RCTL\_SOURCE\_CALLBACK 1**

Receive controller attribute option: Source statistics callback.

### 8.4.3 Typedef Documentation

**8.4.3.1 `typedef int(*) lbmon_ctx_format_deserialize_t(lbm_context_stats_t *Statistics, const char *Source, size_t Length, unsigned short ModuleID, void *FormatClientData)`**

Transform a block of data serialized by the [lbmon\\_ctx\\_format\\_serialize\\_t](#) function into a `lbm_context_stats_t` structure.

**See also:**

[lbmon\\_ctx\\_format\\_serialize\\_t](#)

**Parameters:**

*Statistics* A pointer to an `lbm_context_stats_t` structure into which the data is de-serialized.

*Source* A pointer to a buffer containing the serialized data.

*Length* The length of the serialized data.

*ModuleID* The module ID received in the packet. It may be used to verify and differentiate between different version of the module (and thus the format of the data it expects).

*FormatClientData* A pointer to format-specific client data as returned by the [lbmon\\_format\\_init\\_t](#) function.

**Returns:**

Zero if successful, -1 otherwise. If -1 is returned, the deserialized data will not be delivered to the application.

---

**8.4.3.2 `typedef int(*) lbmmon_ctx_format_serialize_t(char *Destination, size_t *Size, unsigned short *ModuleID, const lbm_context_stats_t *Statistics, void *FormatClientData)`**

This function should transform the `lbm_context_stats_t` structure into a form which can be deserialized by the corresponding `lbmmon_ctx_format_deserialize_t` function.

**See also:**

[lbmmon\\_ctx\\_format\\_deserialize\\_t](#)

**Parameters:**

***Destination*** A pointer to a buffer to receive the serialized format of the `lbm_context_stats_t` statistics.

***Size*** A pointer to a `size_t`. On entry, it will contain the maximum allowed size of the serialized statistics. On exit, it must contain the actual size of the serialized statistics.

***ModuleID*** A pointer to an `unsigned short`, into which the module may write a module identification value. This value is included in the transmitted packet, and may be used by the receiver to verify and differentiate between different version of the module (and thus the format of the data it expects).

***Statistics*** A pointer to the `lbm_context_stats_t` structure to be serialized.

***FormatClientData*** A pointer to format-specific client data as returned by the [lbmmon\\_format\\_init\\_t](#) function.

**Returns:**

Zero if successful, -1 otherwise. If -1 is returned, the serialized data will not be sent.

**8.4.3.3 `typedef void(*) lbmmon_ctx_statistics_cb(const void *AttributeBlock, const lbm_context_stats_t *Statistics, void *ClientData)`**

**Parameters:**

***AttributeBlock*** Pointer to the statistics packet attribute block.

***Statistics*** Pointer to the context statistics.

***ClientData*** Pointer to client-specific data as passed to [lbmmon\\_rctl\\_create\(\)](#).

**8.4.3.4 `typedef struct lbmmon_ctx_statistics_func_t_stct lbmmon_ctx_statistics_func_t`**

A structure used with receive controller options to get/set specific callback information.

**8.4.3.5 `typedef int(*) lbmon_evq_format_deserialize_t(lbm_event_queue_stats_t *Statistics, const char *Source, size_t Length, unsigned short ModuleID, void *FormatClientData)`**

Transform a block of data serialized by the [lbmon\\_evq\\_format\\_serialize\\_t](#) function into a `lbm_event_queue_stats_t` structure.

**See also:**

[lbmon\\_evq\\_format\\_serialize\\_t](#)

**Parameters:**

**Statistics** A pointer to an `lbm_event_queue_stats_t` structure into which the data is deserialized.

**Source** A pointer to a buffer containing the serialized data.

**Length** The length of the serialized data.

**ModuleID** The module ID received in the packet. It may be used to verify and differentiate between different version of the module (and thus the format of the data it expects).

**FormatClientData** A pointer to format-specific client data as returned by the [lbmon\\_format\\_init\\_t](#) function.

**Returns:**

Zero if successful, -1 otherwise. If -1 is returned, the deserialized data will not be delivered to the application.

**8.4.3.6 `typedef int(*) lbmon_evq_format_serialize_t(char *Destination, size_t *Size, unsigned short *ModuleID, const lbm_event_queue_stats_t *Statistics, void *FormatClientData)`**

This function should transform the `lbm_event_queue_stats_t` structure into a form which can be deserialized by the corresponding [lbmon\\_evq\\_format\\_deserialize\\_t](#) function.

**See also:**

[lbmon\\_evq\\_format\\_deserialize\\_t](#)

**Parameters:**

**Destination** A pointer to a buffer to receive the serialized format of the `lbm_event_queue_stats_t` statistics.

**Size** A pointer to a `size_t`. On entry, it will contain the maximum allowed size of the serialized statistics. On exit, it must contain the actual size of the serialized statistics.

**ModuleID** A pointer to an `unsigned short`, into which the module may write a module identification value. This value is included in the transmitted packet, and may be used by the receiver to verify and differentiate between different version of the module (and thus the format of the data it expects).

**Statistics** A pointer to the `lbm_event_queue_stats_t` structure to be serialized.

**FormatClientData** A pointer to format-specific client data as returned by the `lbmmmon_format_init_t` function.

**Returns:**

Zero if successful, -1 otherwise. If -1 is returned, the serialized data will not be sent.

**8.4.3.7 `typedef void(*) lbmmmon_evq_statistics_cb(const void *AttributeBlock, const lbm_event_queue_stats_t *Statistics, void *ClientData)`**

**Parameters:**

**AttributeBlock** Pointer to the statistics packet attribute block.

**Statistics** Pointer to the event queue statistics.

**ClientData** Pointer to client-specific data as passed to `lbmmmon_rctl_create()`.

**8.4.3.8 `typedef struct lbmmmon_evq_statistics_func_t_stct lbmmmon_evq_statistics_func_t`**

A structure used with receive controller options to get/set specific callback information.

**8.4.3.9 `typedef const char*(*) lbmmmon_format_errmsg_t(void)`**

**Returns:**

A string containing a description of the last error encountered by the module.

**8.4.3.10 `typedef int(*) lbmmmon_format_finish_t(void *FormatClientData)`**

Perform any required format module cleanup processing. If format-specific client data was allocated in the `lbmmmon_format_init_t` function, it should be freed in this function.

This function is called by `lbmmmon_sctl_destroy()` and `lbmmmon_rctl_destroy()`.

**Parameters:**

**FormatClientData** A pointer to format-specific client data. This pointer should be freed if it was allocated previously.

**Returns:**

Zero if successful, -1 otherwise.

**8.4.3.11 `typedef int(*) Ibmon_format_init_t(void **FormatClientData, const void *FormatOptions)`**

This function should perform any initialization required by the format module. While, depending on the module, initialization may not be required, representative tasks include allocating a block of format-specific data, parsing options from the supplied options string, and initializing any operating parameters for the module.

This function is called by [lbmon\\_sctl\\_create\(\)](#) and [lbmon\\_rctl\\_create\(\)](#).

**Parameters:**

**FormatClientData** A pointer which may be filled in (by this function) with a pointer to format-specific client data. A pointer to the format-specific data is passed to each function in the module, to be used as the module sees fit.

**FormatOptions** The FormatOptions argument originally passed to [lbmon\\_sctl\\_create\(\)](#) or [lbmon\\_rctl\\_create\(\)](#).

**Returns:**

Zero if successful, -1 otherwise. If -1 is returned, no further calls to the format module will be made, and the calling function ([lbmon\\_sctl\\_create\(\)](#) or [lbmon\\_rctl\\_create\(\)](#)) will return -1.

**8.4.3.12 `typedef struct Ibmon_packet_hdr_t_stct Ibmon_packet_hdr_t`**

A statistics packet consists of four fixed-length and fixed-position fields, as documented below. It is followed by two variable-length fields. The option block is located at packet + sizeof(Ibmon\_packet\_hdr\_t), and is mOptionBlockLength (when properly interpreted) bytes in length (which may be zero). The statistics data block is located immediately following the option block.

---

**8.4.3.13 `typedef int(*) lbmmon_rcv_format_deserialize_t(lbm_rcv_transport_stats_t *Statistics, const char *Source, size_t Length, unsigned short ModuleID, void *FormatClientData)`**

Transform a block of data serialized by the `lbmmon_rcv_format_serialize_t` function into a `lbm_rcv_transport_stats_t` structure.

**See also:**

[lbmmon\\_rcv\\_format\\_serialize\\_t](#)  
[lbmmon\\_src\\_format\\_serialize\\_t](#)  
[lbmmon\\_src\\_format\\_deserialize\\_t](#)

**Parameters:**

*Statistics* A pointer to an `lbm_rcv_transport_stats_t` structure into which the data is deserialized.

*Source* A pointer to a buffer containing the serialized data.

*Length* The length of the serialized data.

*ModuleID* The module ID received in the packet. It may be used to verify and differentiate between different version of the module (and thus the format of the data it expects).

*FormatClientData* A pointer to format-specific client data as returned by the `lbmmon_format_init_t` function.

**Returns:**

Zero if successful, -1 otherwise. If -1 is returned, the deserialized data will not be delivered to the application.

---

**8.4.3.14 `typedef int(*) lbmmon_rcv_format_serialize_t(char *Destination, size_t *Size, unsigned short *ModuleID, const lbm_rcv_transport_stats_t *Statistics, void *FormatClientData)`**

This function should transform the `lbm_rcv_transport_stat_t` structure into a form which can be deserialized by the corresponding `lbmmon_rcv_format_deserialize_t` function.

**See also:**

[lbmmon\\_src\\_format\\_serialize\\_t](#)  
[lbmmon\\_rcv\\_format\\_deserialize\\_t](#)  
[lbmmon\\_src\\_format\\_deserialize\\_t](#)

**Parameters:**

**Destination** A pointer to a buffer to receive the serialized format of the `lbt_rcv_transport_stats_t` statistics.

**Size** A pointer to a `size_t`. On entry, it will contain the maximum allowed size of the serialized statistics. On exit, it must contain the actual size of the serialized statistics.

**ModuleID** A pointer to an `unsigned short`, into which the module may write a module identification value. This value is included in the transmitted packet, and may be used by the receiver to verify and differentiate between different version of the module (and thus the format of the data it expects).

**Statistics** A pointer to the `lbt_rcv_transport_stats_t` structure to be serialized.

**FormatClientData** A pointer to format-specific client data as returned by the [lbmon\\_format\\_init\\_t](#) function.

**Returns:**

Zero if successful, -1 otherwise. If -1 is returned, the serialized data will not be sent.

**8.4.3.15 `typedef void(*) lbmon_rcv_statistics_cb(const void *AttributeBlock, const lbt_rcv_transport_stats_t *Statistics, void *ClientData)`****Parameters:**

**AttributeBlock** Pointer to the statistics packet attribute block.

**Statistics** Pointer to the receiver statistics.

**ClientData** Pointer to client-specific data as passed to [lbmon\\_rctl\\_create\(\)](#).

**8.4.3.16 `typedef struct lbmon_rcv_statistics_func_t_stct lbmon_rcv_statistics_func_t`**

A structure used with receive controller options to get/set specific callback information.

**8.4.3.17 `typedef int(*) lbmon_src_format_deserialize_t(lbt_src_transport_stats_t *Statistics, const char *Source, size_t Length, unsigned short ModuleID, void *FormatClientData)`****See also:**

[lbmon\\_rcv\\_format\\_serialize\\_t](#)  
[lbmon\\_src\\_format\\_serialize\\_t](#)  
[lbmon\\_rcv\\_format\\_deserialize\\_t](#)

**Parameters:**

*Statistics* A pointer to an `lbt_src_transport_stats_t` structure into which the data is deserialized.

*Source* A pointer to a buffer containing the serialized data.

*Length* The length of the serialized data.

*ModuleID* The module ID received in the packet. It may be used to verify and differentiate between different version of the module (and thus the format of the data it expects).

*FormatClientData* A pointer to format-specific client data as returned by the `lbmon_format_init_t` function.

**Returns:**

Zero if successful, -1 otherwise. If -1 is returned, the deserialized data will not be delivered to the application.

**8.4.3.18 `typedef int(*) lbmon_src_format_serialize_t(char *Destination, size_t *Size, unsigned short *ModuleID, const lbt_src_transport_stats_t *Statistics, void *FormatClientData)`**

**See also:**

`lbmon_rcv_format_serialize_t`  
`lbmon_rcv_format_deserialize_t`  
`lbmon_src_format_deserialize_t`

**Parameters:**

*Destination* A pointer to a buffer to receive the serialized format of the `lbt_src_transport_stats_t` statistics.

*Size* A pointer to a `size_t`. On entry, it contains the maximum allowed size of the serialized statistics. On exit, it must contain the actual size of the serialized statistics.

*ModuleID* A pointer to an `unsigned short`, into which the module may write a module identification value. This value is included in the transmitted packet, and may be used by the receiver to verify and differentiate between different version of the module (and thus the format of the data it expects).

*Statistics* A pointer to an `lbt_src_transport_stats_t` structure to be serialized.

*FormatClientData* A pointer to format-specific client data as returned by the `lbmon_format_init_t` function.

**Returns:**

Zero if successful, -1 otherwise. If -1 is returned, the serialized data will not be sent.

**8.4.3.19 `typedef void(*) lbmon_src_statistics_cb(const void *AttributeBlock, const lbm_src_transport_stats_t *Statistics, void *ClientData)`**

**Parameters:**

*AttributeBlock* Pointer to the statistics packet attribute block.

*Statistics* Pointer to the source statistics.

*ClientData* Pointer to client-specific data as passed to [lbmon\\_rctl\\_create\(\)](#).

**8.4.3.20 `typedef struct lbmon_src_statistics_func_t_stct lbmon_src_statistics_func_t`**

A structure used with receive controller options to get/set specific callback information.

**8.4.3.21 `typedef const char*(* lbmon_transport_errmsg_t)(void)`**

**Returns:**

A string containing a description of the last error encountered by the module.

**8.4.3.22 `typedef int(*) lbmon_transport_finishrecv_t(void *TransportClientData)`**

**Parameters:**

*TransportClientData* A pointer to transport-specific client data as returned by the [lbmon\\_transport\\_initrecv\\_t](#) function. If previously allocated by the [lbmon\\_transport\\_initrecv\\_t](#) function, it must be freed in this function.

**Returns:**

Zero if successful, -1 otherwise.

**8.4.3.23 `typedef int(*) lbmon_transport_finishsrc_t(void *TransportClientData)`**

**Parameters:**

*TransportClientData* A pointer to transport-specific client data as returned by the [lbmon\\_transport\\_initsrc\\_t](#) function. If previously allocated by the [lbmon\\_transport\\_initsrc\\_t](#) function, it must be freed in this function.

**Returns:**

Zero if successful, -1 otherwise.

**8.4.3.24 `typedef int(*) lbmmon_transport_initrcv_t(void **TransportClientData, const void *TransportOptions)`****Parameters:**

*TransportClientData* A pointer which may be filled in (by this function) with a pointer to transport-specific client data.

*TransportOptions* The TransportOptions argument originally passed to [lbmmon\\_rctl\\_create\(\)](#).

**Returns:**

Zero if successful, -1 otherwise.

**8.4.3.25 `typedef int(*) lbmmon_transport_initsrc_t(void **TransportClientData, const void *TransportOptions)`****Parameters:**

*TransportClientData* A pointer which may be filled in (by this function) with a pointer to transport-specific client data.

*TransportOptions* The TransportOptions argument originally passed to [lbmmon\\_sctl\\_create\(\)](#).

**Returns:**

Zero if successful, -1 otherwise.

**8.4.3.26 `typedef int(*) lbmmon_transport_receive_t(char *Data, size_t *Length, unsigned int TimeoutMS, void *TransportClientData)`****Parameters:**

*Data* Pointer to a buffer into which the received (serialized) data is to be placed.

*Length* Pointer to a size\_t variable. On entry, it contains the maximum number of bytes to read. On exit, it must contain the actual number of bytes read.

*TimeoutMS* Maximum time, in milliseconds, the function may wait for incoming data before returning a timeout indicator.

*TransportClientData* A pointer to transport-specific client data as returned by the [lbmmon\\_transport\\_initrcv\\_t](#) function.

**Returns:**

Zero if successful, >0 if timeout exceeded, -1 otherwise.

**8.4.3.27 `typedef int(*) lbmon_transport_send_t(const char *Data, size_t Length, void *TransportClientData)`****Parameters:**

*Data* Pointer to the serialized statistics data.

*Length* Length of the serialized statistics data.

*TransportClientData* A pointer to transport-specific client data as returned by the [lbmon\\_transport\\_initsrc\\_t](#) function.

**Returns:**

Zero if successful, -1 otherwise.

## 8.4.4 Function Documentation

**8.4.4.1 LBMEExpDLL int lbmon\_attr\_get\_appsourceid (const void \* AttributeBlock, char \* ApplicationSourceID, size\_t Length)****Parameters:**

*AttributeBlock* Pointer to the attribute block as passed to the callback function.

*ApplicationSourceID* Pointer to a buffer to receive the application source ID as a null-terminated string.

*Length* Maximum length of *ApplicationSourceID*

**Returns:**

0 if successful, -1 if the attribute does not exist.

**8.4.4.2 LBMEExpDLL int lbmon\_attr\_get\_contextid (const void \* AttributeBlock, lbm\_ulong\_t \* ContextID)****Parameters:**

*AttributeBlock* Pointer to the attribute block as passed to the callback function.

*ContextID* Pointer to a variable to receive the context ID.

**Returns:**

0 if successful, -1 if the attribute does not exist.

**Deprecated**

Use [lbmmmon\\_attr\\_get\\_objectid](#) instead.

**8.4.4.3 LBMExpDLL int lbmmmon\_attr\_get\_ipv4sender (const void \*  
AttributeBlock, lbm\_uint\_t \* Address)****Parameters:**

*AttributeBlock* Pointer to the attribute block as passed to the callback function.

*Address* Pointer to a 32-bit integer to receive the IPV4 address in network order.

**Returns:**

0 if successful, -1 if the attribute does not exist.

**8.4.4.4 LBMExpDLL int lbmmmon\_attr\_get\_objectid (const void \*  
AttributeBlock, lbm\_ulong\_t \* ObjectID)****Parameters:**

*AttributeBlock* Pointer to the attribute block as passed to the callback function.

*ObjectID* Pointer to a variable to receive the object ID.

**Returns:**

0 if successful, -1 if the attribute does not exist.

**8.4.4.5 LBMExpDLL int lbmmmon\_attr\_get\_processid (const void \*  
AttributeBlock, lbm\_ulong\_t \* ProcessID)****Parameters:**

*AttributeBlock* Pointer to the attribute block as passed to the callback function.

*ProcessID* Pointer to a variable to receive the process ID.

**Returns:**

0 if successful, -1 if the attribute does not exist.

**8.4.4.6 LBMEExpDLL int lbmon\_attr\_get\_source (const void \* *AttributeBlock*,  
lbm\_ulong\_t \* *Source*)****Parameters:**

***AttributeBlock*** Pointer to the attribute block as passed to the callback function.

***Source*** Pointer to a variable to receive the source flag.

**Returns:**

0 if successful, -1 if the attribute does not exist.

**8.4.4.7 LBMEExpDLL int lbmon\_attr\_get\_timestamp (const void \*  
*AttributeBlock*, time\_t \* *Timestamp*)****Parameters:**

***AttributeBlock*** Pointer to the attribute block as passed to the callback function.

***Timestamp*** Pointer to a `time_t` to receive the timestamp.

**Returns:**

0 if successful, -1 if the attribute does not exist.

**8.4.4.8 LBMEExpDLL int lbmon\_context\_monitor (lbmon\_sctl\_t \* *Control*,  
lbm\_context\_t \* *Context*, const char \* *ApplicationSourceID*, unsigned int  
*Seconds*)**

When a context is monitored, statistics are gathered for all transports on that context, broken out by transport. Monitoring may be done at regular intervals, specified by the `Seconds` parameter. As an alternative, passing zero for `Seconds` will not automatically monitor the context, but instead require an explicit call to [lbmon\\_sctl\\_sample\(\)](#).

If monitoring is to be used as a form of heartbeat, the preferred method is to call [lbmon\\_sctl\\_sample\(\)](#) from a context thread or event queue timer callback. This ensures that the object actually processing the messages is the one generating the monitoring statistics, guaranteeing that it is truly acting as a heartbeat mechanism.

**Parameters:**

***Control*** Pointer to an `lbmon_sctl_t` which is to be used to monitor the context.

***Context*** Pointer to an `lbm_context_t` which will be monitored.

***ApplicationSourceID*** Null-terminated string containing an application-specified source identifier. If a NULL pointer or an empty string is passed, the application name will be used.

**Seconds** Interval (in seconds) at which monitoring information will be gathered and sent. If zero, the context will not be automatically monitored, but instead will be monitored upon a call to lbmmon\_ctl\_sample().

**Returns:**

Zero if successful, -1 otherwise.

**8.4.4.9 LBMExpDLL int lbmmon\_context\_unmonitor (lbmmon\_sctl\_t \* Control, lbm\_context\_t \* Context)**

Unregister a context to prevent further monitoring of that context.

**Parameters:**

**Control** Pointer to an lbmmon\_sctl\_t which is used to monitor the context.

**Context** Pointer to an previously registered lbm\_context\_t.

**Returns:**

Zero if successful, -1 otherwise.

**8.4.4.10 LBMExpDLL const char\* lbmmon\_errmsg (void)****Returns:**

A pointer to a static character array containing the last error message.

**8.4.4.11 LBMExpDLL int lbmmon\_errnum (void)****Returns:**

The last error encountered. See LBMMON\_ERR\_\*.

**8.4.4.12 LBMExpDLL int lbmmon\_evq\_monitor (lbmmon\_sctl\_t \* Control, lbm\_event\_queue\_t \* EventQueue, const char \* ApplicationSourceID, unsigned int Seconds)**

Monitoring may be done at regular intervals, specified by the *Seconds* parameter. As an alternative, passing zero for *Seconds* will not automatically monitor the receiver, but instead require an explicit call to lbmmon\_ctl\_sample().

If monitoring is to be used as a form of heartbeat, the preferred method is to call `lbmon_ctl_sample()` from a context thread or event queue timer callback. This ensures that the object actually processing the messages is the one generating the monitoring statistics, guaranteeing that it is truly acting as a heartbeat mechanism.

**Parameters:**

***Control*** Pointer to an `lbmon_sctl_t` which is to be used to monitor the context.

***Receiver*** Pointer to an `lbt_event_queue_t` which will be monitored.

***ApplicationSourceID*** Null-terminated string containing an application-specified source identifier. If a NULL pointer or an empty string is passed, the application name will be used.

***Seconds*** Interval (in seconds) at which monitoring information will be gathered and sent. If zero, the receiver will not be automatically monitored, but instead will be monitored upon a call to `lbmon_sctl_sample()`.

**Returns:**

Zero if successful, -1 otherwise.

**8.4.4.13 LBMDLL int lbmon\_evq\_unmonitor (`lbmon_sctl_t * Control,`  
`lbt_event_queue_t * EventQueue)`**

Unregister an event queue to prevent further monitoring of that receiver.

**Parameters:**

***Control*** Pointer to an `lbmon_sctl_t` which is used to monitor the context.

***EventQueue*** Pointer to an previously registered `lbt_event_queue_t`.

**Returns:**

Zero if successful, -1 otherwise.

**8.4.4.14 const char\* lbmon\_next\_key\_value\_pair (`const char * String, char *`  
`Key, size_t KeySize, char * Value, size_t ValueSize)`**

This is a convenience utility function to facilitate processing of a semicolon-separated list of key/value pairs. Each pair is of the form "key=value".

**Parameters:**

***String*** A pointer to the unprocessed part of the semicolon-separated list.

**Key** Pointer to a character string into which is written the null-terminated key.

**KeySize** Maximum length of *Key*.

**Value** Pointer to a character string into which is written the null-terminated value.

**ValueSize** Maximum length of *Value*.

**Returns:**

NULL if no key/value pair is found. Otherwise a pointer to the remainder of the string, to be passed to subsequent calls to [lbmmmon\\_next\\_key\\_value\\_pair\(\)](#).

#### 8.4.4.15 LBMExpDLL int lbmmmon\_rctl\_attr\_create (lbmmmon\_rctl\_attr\_t \*\* *Attributes*)

The attribute object is created and initialized.

**Parameters:**

**Attributes** A pointer to a pointer to an LBMMON receive controller attribute structure. Will be filled in by the function to point to the newly created lbmmmon\_rctl\_attr\_t object.

**Returns:**

0 if successful, -1 otherwise.

#### 8.4.4.16 LBMExpDLL int lbmmmon\_rctl\_attr\_delete (lbmmmon\_rctl\_attr\_t \* *Attributes*)

The attribute object is cleaned up and deleted.

**Parameters:**

**Attributes** A pointer to an LBMMON receive controller attribute structure as created by [lbmmmon\\_rctl\\_attr\\_create](#).

**Returns:**

0 if successful, -1 otherwise.

**8.4.4.17 LBMEExpDLL int lbmon\_rctl\_attr\_getopt (lbmon\_rctl\_attr\_t \*  
Attributes, int Option, void \* Value, size\_t \* Length)****Parameters:**

*Attributes* The attributes object to get the option value for.

*Option* The option to get. See LBMMON\_RCTL\_ATTR\_\*

*Value* Pointer to the option value structure to be filled.

*Length* Length (in bytes) of the *Value* structure when passed in. Upon return, this is set to the actual size of the *Value* structure filled in.

**Returns:**

0 if successful, -1 otherwise.

**8.4.4.18 LBMEExpDLL int lbmon\_rctl\_attr\_setopt (lbmon\_rctl\_attr\_t \*  
Attributes, int Option, void \* Value, size\_t Length)****Parameters:**

*Attributes* The attributes object to set the option value for.

*Option* The option to set. See LBMMON\_RCTL\_ATTR\_\*

*Value* The value to set for the option.

*Length* The size (in bytes) of *Value*.

**Returns:**

0 if successful, -1 otherwise.

**8.4.4.19 LBMEExpDLL int lbmon\_rctl\_create (lbmon\_rctl\_t \*\* Control,  
const lbmon\_format\_func\_t \* Format, const void \* FormatOptions,  
const lbmon\_transport\_func\_t \* Transport, const void \*  
TransportOptions, lbmon\_rctl\_attr\_t \* Attributes, void \* ClientData)**

This creates an instance of an LBM Monitoring Receive Controller.

**Parameters:**

*Control* A pointer to a pointer to an LBM Monitoring Receive Control object. Will be filled in by this function to point to the newly created lbmon\_rctl\_t object.

*Format* A pointer to an lbmon\_format\_func\_t object which has been filled in with the appropriate function pointers.

**FormatOptions** A block of data which is passed to the format module's initialization function. This may be used to pass configuration options to the format module.

**Transport** A pointer to an lbmmmon\_transport\_func\_t object which has been filled in with the appropriate function pointers.

**TransportOptions** A block of data which is passed to the transport module's initialization function. This may be used to pass configuration options to the transport module.

**Attributes** A pointer to an lbmmmon\_rctl\_attr\_t object which has been filled in with the appropriate options.

**ClientData** A pointer to a block of memory which can be used for client-specific data. It is passed to all callback functions.

**Returns:**

0 if successful, -1 otherwise.

#### 8.4.4.20 LBMExpDLL int lbmmmon\_rctl\_destroy (lbmmmon\_rctl\_t \* *Control*)

Destroys a monitoring controller.

**Parameters:**

**Control** Pointer to an lbmmmon\_rctl\_t to be destroyed.

**Returns:**

Zero if successful, -1 otherwise.

#### 8.4.4.21 LBMExpDLL int lbmmmon\_rev\_monitor (lbmmmon\_sctl\_t \* *Control*, lbm\_rev\_t \* *Receiver*, const char \* *ApplicationSourceID*, unsigned int *Seconds*)

Monitoring may be done at regular intervals, specified by the *Seconds* parameter. As an alternative, passing zero for *Seconds* will not automatically monitor the receiver, but instead require an explicit call to lbmmmon\_ctl\_sample().

If monitoring is to be used as a form of heartbeat, the preferred method is to call lbmmmon\_ctl\_sample() from a context thread or event queue timer callback. This ensures that the object actually processing the messages is the one generating the monitoring statistics, guaranteeing that it is truly acting as a heartbeat mechanism.

**Parameters:**

**Control** Pointer to an lbmmmon\_sctl\_t which is to be used to monitor the context.

**Receiver** Pointer to an `lbt_rcv_t` which will be monitored.

**ApplicationSourceID** Null-terminated string containing an application-specified source identifier. If a NULL pointer or an empty string is passed, the application name will be used.

**Seconds** Interval (in seconds) at which monitoring information will be gathered and sent. If zero, the receiver will not be automatically monitored, but instead will be monitored upon a call to `lbmon_sctl_sample()`.

**Returns:**

Zero if successful, -1 otherwise.

#### 8.4.4.22 LBMEExpDLL int lbmon\_rcv\_unmonitor (`lbmon_sctl_t * Control,` `lbt_rcv_t * Receiver`)

Unregister a receiver to prevent further monitoring of that receiver.

**Parameters:**

**Control** Pointer to an `lbmon_sctl_t` which is used to monitor the context.

**Receiver** Pointer to an previously registered `lbt_rcv_t`.

**Returns:**

Zero if successful, -1 otherwise.

#### 8.4.4.23 LBMEExpDLL int lbmon\_sctl\_create (`lbmon_sctl_t ** Control,` `const lbmon_format_func_t * Format, const void * FormatOptions,` `const lbmon_transport_func_t * Transport, const void *` `TransportOptions)`

This creates an instance of an LBM Monitoring Source Controller.

**Parameters:**

**Control** A pointer to a pointer to an LBM Monitoring Source Control object. It will be filled in by this function to point to the newly created `lbmon_sctl_t` object.

**Format** A pointer to an `lbmon_format_func_t` object which has been filled in with the appropriate function pointers.

**FormatOptions** A block of data which is passed to the format module's initialization function. This may be used to pass configuration options to the format module.

**Transport** A pointer to an lbmmmon\_transport\_func\_t object which has been filled in with the appropriate function pointers.

**TransportOptions** A block of data which is passed to the transport module's initialization function. This may be used to pass configuration options to the transport module.

**Returns:**

0 if successful, -1 otherwise.

#### 8.4.4.24 LBMExpDLL int lbmmmon\_sctl\_destroy (lbmmmon\_sctl\_t \* *Control*)

Destroys a monitoring controller. Any contexts, sources, or receivers currently registered to the controller will be automatically unregistered.

**Parameters:**

*Control* Pointer to an lbmmmon\_sctl\_t to be destroyed.

**Returns:**

Zero if successful, -1 otherwise.

#### 8.4.4.25 LBMExpDLL int lbmmmon\_sctl\_sample (lbmmmon\_sctl\_t \* *Control*)

**Parameters:**

*Control* Pointer to an existing lbmmmon\_sctl\_t controller.

**Returns:**

Zero if successful, -1 otherwise.

#### 8.4.4.26 LBMExpDLL int lbmmmon\_sctl\_sample\_ex (lbmmmon\_sctl\_t \* *Control*, const char \* *ApplicationSourceID*)

**Parameters:**

*Control* Pointer to an existing lbmmmon\_sctl\_t controller.

*ApplicationSourceID* Null-terminated string containing an application-specified source identifier. This overrides any application source ID passed to any of the lbmmmon\_\*\_monitor() functions for this call only. If a NULL pointer or an empty string is passed, the original application source ID will be used.

**Returns:**

Zero if successful, -1 otherwise.

#### 8.4.4.27 LBMEExpDLL int lbmon\_src\_monitor (lbmon\_sctl\_t \* *Control*, lbt\_src\_t \* *Source*, const char \* *ApplicationSourceID*, unsigned int *Seconds*)

Monitoring may be done at regular intervals, specified by the *Seconds* parameter. As an alternative, passing zero for *Seconds* will not automatically monitor the source, but instead require an explicit call to [lbmon\\_sctl\\_sample\(\)](#).

If monitoring is to be used as a form of heartbeat, the preferred method is to call [lbmon\\_ctl\\_sample\(\)](#) from a context thread or event queue timer callback. This ensures that the object actually processing the messages is the one generating the monitoring statistics, guaranteeing that it is truly acting as a heartbeat mechanism.

##### Parameters:

***Control*** Pointer to an [lbmon\\_sctl\\_t](#) which is to be used to monitor the context.

***Source*** Pointer to an [lbt\\_src\\_t](#) which will be monitored.

***ApplicationSourceID*** Null-terminated string containing an application-specified source identifier. If a NULL pointer or an empty string is passed, the application name will be used.

***Seconds*** Interval (in seconds) at which monitoring information will be gathered and sent. If zero, the source will not be automatically monitored, but instead will be monitored upon a call to [lbmon\\_sctl\\_sample\(\)](#).

##### Returns:

Zero if successful, -1 otherwise.

#### 8.4.4.28 LBMEExpDLL int lbmon\_src\_unmonitor (lbmon\_sctl\_t \* *Control*, lbt\_src\_t \* *Source*)

Unregister a source to prevent further monitoring of that source.

##### Parameters:

***Control*** Pointer to an [lbmon\\_sctl\\_t](#) which is used to monitor the context.

***Source*** Pointer to a previously registered [lbt\\_src\\_t](#).

##### Returns:

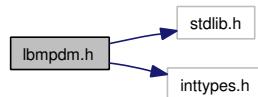
Zero if successful, -1 otherwise.

## 8.5 lbmpdm.h File Reference

Ultra Messaging (UM) Pre-Defined Message (PDM) API.

```
#include <stdlib.h>
#include <inttypes.h>
```

Include dependency graph for lbmpdm.h:



### Data Structures

- struct **lbmpdm\_decimal\_t**  
*Structure to hold a scaled decimal number. A scaled decimal number consists of a mantissa m and an exponent exp. It represents the value  $m \cdot 10^{exp}$ .*
- struct **lbmpdm\_timestamp\_t**  
*Structure to hold a timestamp value.*
- struct **lbmpdm\_field\_info\_attr\_stct\_t**  
*Attribute struct to be passed along with the name when adding field info to a definition.*
- struct **lbmpdm\_field\_value\_stct\_t**  
*Field value struct that can be populated with a field value when passed to the `lbmpdm_msg_get_field_value_stct` function.*

### Defines

- #define **LBMPDMDLL**  
*PDM API function return codes.*
- #define **PRIuSZ** "zu"
- #define **SCNuSZ** "zu"
- #define **PDM\_FALSE** (uint8\_t) 0
- #define **PDM\_TRUE** (uint8\_t) 1
- #define **PDM\_FIELD\_INFO\_FLAG\_REQ** 0x1
- #define **PDM\_FIELD\_INFO\_FLAG\_FIXED\_STR\_LEN** 0x2
- #define **PDM\_FIELD\_INFO\_FLAG\_NUM\_ARR\_ELEM** 0x4

- #define PDM\_MSG\_FLAG\_VAR\_OR\_OPT\_FLDS\_SET 0x1
- #define PDM\_MSG\_FLAG\_INCL\_DEFN 0x2
- #define PDM\_MSG\_FLAG\_USE\_MSG\_DEFN\_IF\_NEEDED 0x4
- #define PDM\_MSG\_FLAG\_TRY\_LOAD\_DEFN\_FROM\_CACHE 0x8
- #define PDM\_MSG\_FLAG\_NEED\_BYTE\_SWAP 0x10
- #define PDM\_MSG\_FLAG\_DEL\_DEFN\_WHEN\_REPLACED 0x20
- #define PDM\_MSG\_VER\_POLICY\_EXACT 0
- #define PDM\_MSG\_VER\_POLICY\_BEST 1
- #define PDM\_SUCCESS 0
- #define PDM\_FAILURE -1
- #define PDM\_ERR\_FIELD\_IS\_NULL 1
- #define PDM\_ERR\_NO\_MORE\_FIELDS 2
- #define PDM\_ERR\_INSUFFICIENT\_BUFFER\_LENGTH 3
- #define PDM\_ERR\_EINVAL 4
- #define PDM\_ERR\_FIELD\_NOT\_FOUND 5
- #define PDM\_ERR\_MSG\_INVALID 6
- #define PDM\_ERR\_DEFN\_INVALID 7
- #define PDM\_ERR\_NOMEM 8
- #define PDM\_ERR\_REQ\_FIELD\_NOT\_SET 9
- #define PDM\_ERR\_CREATE\_SECTION 10
- #define PDM\_ERR\_CREATE\_BUFFER 11
- #define PDM\_INTERNAL\_TYPE\_INVALID -1
- #define PDM\_TYPE\_BOOLEAN 0
- #define PDM\_TYPE\_INT8 1
- #define PDM\_TYPE\_UINT8 2
- #define PDM\_TYPE\_INT16 3
- #define PDM\_TYPE\_UINT16 4
- #define PDM\_TYPE\_INT32 5
- #define PDM\_TYPE\_UINT32 6
- #define PDM\_TYPE\_INT64 7
- #define PDM\_TYPE\_UINT64 8
- #define PDM\_TYPE\_FLOAT 9
- #define PDM\_TYPE\_DOUBLE 10
- #define PDM\_TYPE\_DECIMAL 11
- #define PDM\_TYPE\_TIMESTAMP 12
- #define PDM\_TYPE\_FIX\_STRING 13
- #define PDM\_TYPE\_STRING 14
- #define PDM\_TYPE\_FIX\_UNICODE 15
- #define PDM\_TYPE\_UNICODE 16
- #define PDM\_TYPE\_BLOB 17
- #define PDM\_TYPE\_MESSAGE 18
- #define PDM\_TYPE\_BOOLEAN\_ARR 19
- #define PDM\_TYPE\_INT8\_ARR 20

- #define PDM\_TYPE\_UINT8\_ARR 21
- #define PDM\_TYPE\_INT16\_ARR 22
- #define PDM\_TYPE\_UINT16\_ARR 23
- #define PDM\_TYPE\_INT32\_ARR 24
- #define PDM\_TYPE\_UINT32\_ARR 25
- #define PDM\_TYPE\_INT64\_ARR 26
- #define PDM\_TYPE\_UINT64\_ARR 27
- #define PDM\_TYPE\_FLOAT\_ARR 28
- #define PDM\_TYPE\_DOUBLE\_ARR 29
- #define PDM\_TYPE\_DECIMAL\_ARR 30
- #define PDM\_TYPE\_TIMESTAMP\_ARR 31
- #define PDM\_TYPE\_FIX\_STRING\_ARR 32
- #define PDM\_TYPE\_STRING\_ARR 33
- #define PDM\_TYPE\_FIX\_UNICODE\_ARR 34
- #define PDM\_TYPE\_UNICODE\_ARR 35
- #define PDM\_TYPE\_BLOB\_ARR 36
- #define PDM\_TYPE\_MESSAGE\_ARR 37
- #define PDM\_DEFN\_STR\_FIELD\_NAMES 0
- #define PDM\_DEFN\_INT\_FIELD\_NAMES 1
- #define PDM\_ITER\_INVALID\_FIELD\_HANDLE -1

## Typedefs

- typedef int32\_t **lbmpdm\_field\_handle\_t**

*Type representing a handle to a message field. Field handles are returned when adding a field to a definition, or can be retrieved from a definition using the `lbmpdm_get_field_handle_by_str_name` or `lbpdm_get_field_handle_by_int_name` functions.*
- typedef lbmpdm\_msg\_stct\_t **lbmpdm\_msg\_t**

*Structure to hold a pdm message.*
- typedef lbmpdm\_defn\_stct\_t **lbmpdm\_defn\_t**

*Structure to hold a pdm definition.*
- typedef **lbmpdm\_field\_info\_attr\_stct\_t** **lbmpdm\_field\_info\_attr\_t**
- typedef **lbmpdm\_field\_value\_stct\_t** **lbmpdm\_field\_value\_t**
- typedef lbmpdm\_iter\_stct\_t **lbmpdm\_iter\_t**

*Iterator structure that is used to traverse the fields of a message.*

## Functions

- LBMPDMDLL int [lbmpdm\\_errnum](#) ()  
*Return the error number last encountered by this thread.*
- LBMPDMDLL const char \* [lbmpdm\\_errmsg](#) ()  
*Return an ASCII string containing the error message last encountered by this thread.*
- LBMPDMDLL int [lbmpdm\\_cache\\_init](#) (uint32\_t cache\_size)  
*initialize the cache for a given number of buckets. If 0 is given the the cache will default. The default can be altered via a PDM attribute.*
- LBMPDMDLL int [lbmpdm\\_cache\\_struct\\_add](#) ([lbmpdm\\_defn\\_t](#) \*defn)  
*add a definition structure to the cache. It is assumed that the structure has a unique id, if the id is set to zero the structure will not be inserted into the map.*
- LBMPDMDLL int [lbmpdm\\_cache\\_struct\\_remove](#) (int32\_t id)  
*delete a definition structure from the cache. Does not error if the structure doesn't exist.*
- LBMPDMDLL int [lbmpdm\\_cache\\_struct\\_remove\\_by\\_version](#) (int32\_t id, uint8\_t vers\_major, uint8\_t vers\_minor)  
*delete a definition structure from the cache by its id and version. Does not error if the structure doesn't exist.*
- LBMPDMDLL int [lbmpdm\\_cache\\_struct\\_find](#) ([lbmpdm\\_defn\\_t](#) \*\*defn, int32\_t id)  
*find a given definition structure by id and return the structure for it. Returns PDM\_FAILURE for nothing found, and PDM\_SUCCESS for something found. Note, since a structure with the id of 0 won't exist in the cache you will never find one being returned from this find with an id of 0.*
- LBMPDMDLL int [lbmpdm\\_cache\\_struct\\_find\\_by\\_version](#) ([lbmpdm\\_defn\\_t](#) \*\*defn, int32\_t id, uint8\_t vers\_major, uint8\_t vers\_minor)  
*find a given definition structure by id, major version, and minor version and return the structure for it. Returns PDM\_FAILURE for nothing found, and PDM\_SUCCESS for something found. Note, since a structure with the id of 0 won't exist in the cache you will never find one being returned from this find with an id of 0.*
- LBMPDMDLL int [lbmpdm\\_cache\\_clear\\_all](#) ()  
*nuke the whole cache, this deletes all the structures within the cache as well.*
- LBMPDMDLL [lbmpdm\\_field\\_handle\\_t](#) [lbmpdm\\_defn\\_get\\_field\\_handle\\_by\\_str\\_name](#) ([lbmpdm\\_defn\\_t](#) \*defn, const char \*str\_name)

*Retrieve a field handle from a definition via name.*

- LBMPDMDExpDLL [lbmpdm\\_field\\_handle\\_t](#) [lbmpdm\\_defn\\_get\\_field\\_handle\\_by\\_int\\_name](#) ([lbmpdm\\_defn\\_t](#) \*defn, int32\_t int\_name)

*Retrieve a field handle from a definition via name.*

- LBMPDMDExpDLL int [lbmpdm\\_defn\\_create](#) ([lbmpdm\\_defn\\_t](#) \*\*defn, int32\_t num\_fields, int32\_t id, int8\_t vrs\_mjr, int8\_t vrs\_mnr, uint8\_t field\_names\_type)

*Create a definition, with the passed number of fields. The num\_fields is required to be at least 1. The number of fields can grow beyond this value, it is used initially to size the internal field info array.*

- LBMPDMDExpDLL int [lbmpdm\\_defn\\_delete](#) ([lbmpdm\\_defn\\_t](#) \*defn)

*delete a given definition.*

- LBMPDMDExpDLL int [lbmpdm\\_defn\\_finalize](#) ([lbmpdm\\_defn\\_t](#) \*defn)

*make this definition final. This needs to be done before using it in a message.*

- LBMPDMDExpDLL [lbmpdm\\_field\\_handle\\_t](#) [lbmpdm\\_defn\\_add\\_field\\_info\\_by\\_str\\_name](#) ([lbmpdm\\_defn\\_t](#) \*defn, const char \*str\_name, int16\_t type, [lbmpdm\\_field\\_info\\_attr\\_t](#) \*info\_attr)

*adds field info to the definition by string name*

- LBMPDMDExpDLL [lbmpdm\\_field\\_handle\\_t](#) [lbmpdm\\_defn\\_add\\_field\\_info\\_by\\_int\\_name](#) ([lbmpdm\\_defn\\_t](#) \*defn, int32\_t int\_name, int16\_t type, [lbmpdm\\_field\\_info\\_attr\\_t](#) \*info\_attr)

*adds field info to the definition by integer name*

- LBMPDMDExpDLL uint32\_t [lbmpdm\\_defn\\_get\\_length](#) ([lbmpdm\\_defn\\_t](#) \*defn)

*Gets the exact length of the serialized defn. This can be used to allocate a buffer of the exact length needed to serialize the defn.*

- LBMPDMDExpDLL int32\_t [lbmpdm\\_defn\\_get\\_id](#) ([lbmpdm\\_defn\\_t](#) \*defn)

*Gets the id of the definition.*

- LBMPDMDExpDLL int32\_t [lbmpdm\\_defn\\_get\\_num\\_fields](#) ([lbmpdm\\_defn\\_t](#) \*defn)

*Gets the number of fields in the definition.*

- LBMPDMDExpDLL int8\_t [lbmpdm\\_defn\\_get\\_msg\\_vers\\_major](#) ([lbmpdm\\_defn\\_t](#) \*defn)

*Gets the message major version number from the definition.*

- LBMPDMDExpDLL int8\_t `lbmpdm_defn_get_msg_vers_minor` (`lbmpdm_defn_t` \*`defn`)

*Gets the message minor version number from the definition.*
- LBMPDMDExpDLL uint8\_t `lbmpdm_defn_get_field_names_type` (`lbmpdm_defn_t` \*`defn`)

*Gets the field names type (either PDM\_DEFN\_STR\_FIELD\_NAMES or PDM\_DEFN\_INT\_FIELD\_NAMES) from the definition.*
- LBMPDMDExpDLL uint8\_t `lbmpdm_defn_is_finalized` (`lbmpdm_defn_t` \*`defn`)

*Gets whether or not the definition has been finalized (either PDM\_TRUE or PDM\_FALSE).*
- LBMPDMDExpDLL const char \* `lbmpdm_defn_get_field_info_str_name` (`lbmpdm_defn_t` \*`defn`, `lbmpdm_field_handle_t` `handle`)

*Gets the string field name from a given definition's field handle.*
- LBMPDMDExpDLL int32\_t `lbmpdm_defn_get_field_info_int_name` (`lbmpdm_defn_t` \*`defn`, `lbmpdm_field_handle_t` `handle`)

*Gets the integer field name from a given definition's field handle.*
- LBMPDMDExpDLL int16\_t `lbmpdm_defn_get_field_info_type` (`lbmpdm_defn_t` \*`defn`, `lbmpdm_field_handle_t` `handle`)

*Gets the PDM field type from a given definition's field handle.*
- LBMPDMDExpDLL int `lbmpdm_defn_serialize` (`lbmpdm_defn_t` \*`defn`, char \*`buffer`, uint32\_t \*`defn_len`)

*Serialize a defn to a buffer. In normal usage this is not needed as the defn is either known in advance or sent as part of a message. The defn that is passed in is serialized into the caller's supplied buffer.*
- LBMPDMDExpDLL int `lbmpdm_defn_deserialize` (`lbmpdm_defn_t` \*`defn`, const char \*`bufptr`, uint32\_t `buflen`, uint8\_t `swap_bytes`)

*Deserialize the associated buffer into a newly created defn.*
- LBMPDMDExpDLL int `lbmpdm_msg_create` (`lbmpdm_msg_t` \*\*`message`, `lbmpdm_defn_t` \*`defn`, uint32\_t `flags`)

*creates a message with the specified definition*
- LBMPDMDExpDLL int `lbmpdm_msg_delete` (`lbmpdm_msg_t` \*`message`)

*Delete an lbmpdm\_msg\_t object and all associated resources (except the defn) This deletes a previously created PDM message and all resources associated with the message.*

- LBMPDMDLL int lbmpdm\_msg\_and\_defn\_delete (lbmpdm\_msg\_t \*message)

*Delete an lbmpdm\_msg\_t object and all associated resources (including the defn) This deletes a previously created PDM message and all resources associated with the message.*

- LBMPDMDLL uint32\_t lbmpdm\_msg\_get\_length (const lbmpdm\_msg\_t \*message)

*Gets the exact length of the serialized message. This can be used to allocate a buffer of the exact length needed to serialize the message.*

- LBMPDMDLL lbmpdm\_defn\_t \* lbmpdm\_msg\_get\_defn (const lbmpdm\_msg\_t \*message)

*Gets a pointer to the message definition.*

- LBMPDMDLL uint8\_t lbmpdm\_msg\_is\_field\_set (lbmpdm\_msg\_t \*message, lbmpdm\_field\_handle\_t handle)

*Gets whether or not the field value has been set.*

- LBMPDMDLL int lbmpdm\_msg\_get\_field\_value\_stct (lbmpdm\_msg\_t \*message, lbmpdm\_field\_handle\_t handle, lbmpdm\_field\_value\_t \*field\_value)

*Populates a field value struct with the value from the message.*

- LBMPDMDLL int lbmpdm\_msg\_get\_field\_value (lbmpdm\_msg\_t \*message, lbmpdm\_field\_handle\_t handle, void \*value, size\_t \*len)

*Gets a field value from the message.*

- LBMPDMDLL int lbmpdm\_msg\_get\_field\_value\_vec (lbmpdm\_msg\_t \*message, lbmpdm\_field\_handle\_t handle, void \*value, size\_t len[], size\_t \*num\_arr\_elem)

*Gets an array of field values from the message.*

- LBMPDMDLL int lbmpdm\_msg\_set\_field\_value (lbmpdm\_msg\_t \*message, lbmpdm\_field\_handle\_t handle, void \*value, size\_t len)

*Sets a field value in a message.*

- LBMPDMDLL int lbmpdm\_msg\_set\_field\_value\_vec (lbmpdm\_msg\_t \*message, lbmpdm\_field\_handle\_t handle, void \*value, size\_t len[], size\_t num\_arr\_elem)

*Sets an array of field values in a message.*

- LBMPDMDLL int [lbmpdm\\_msg\\_remove\\_field\\_value](#) ([lbmpdm\\_msg\\_t](#) \*message, [lbmpdm\\_field\\_handle\\_t](#) handle)

*Removes a field value from a message (marking it unset).*

- LBMPDMDLL int [lbmpdm\\_msg\\_set\\_incl\\_defn\\_flag](#) ([lbmpdm\\_msg\\_t](#) \*message)

*Sets the message include definition flag.*

- LBMPDMDLL int [lbmpdm\\_msg\\_unset\\_incl\\_defn\\_flag](#) ([lbmpdm\\_msg\\_t](#) \*message)

*Unsets the message include definition flag.*

- LBMPDMDLL int [lbmpdm\\_field\\_value\\_stct\\_delete](#) ([lbmpdm\\_field\\_value\\_t](#) \*field\_value)

*Deletes the allocated resources inside the field value struct. This does NOT free the actual field value struct passed in (which should be done outside PDM). Also, this does not affect the field value in the message, only this field value struct.*

- LBMPDMDLL int [lbmpdm\\_msg\\_serialize](#) ([lbmpdm\\_msg\\_t](#) \*message, char \*buffer)

*Serialize a message to a buffer. The message that is passed in is serialized into the caller's supplied buffer.*

- LBMPDMDLL int [lbmpdm\\_msg\\_deserialize](#) ([lbmpdm\\_msg\\_t](#) \*message, const char \*bufptr, uint32\_t buflen)

*Deserialize the associated buffer into a newly created message.*

- LBMPDMDLL char \* [lbmpdm\\_msg\\_get\\_data](#) ([lbmpdm\\_msg\\_t](#) \*message)

*Serialize a message to a buffer and return the buffer. The message that is passed in is serialized into a buffer which will be cleaned up when the message is deleted. Use [lbmpdm\\_msg\\_get\\_length](#) to get the length of the buffer.*

- LBMPDMDLL int [lbmpdm\\_iter\\_create](#) ([lbmpdm\\_iter\\_t](#) \*\*iter, [lbmpdm\\_msg\\_t](#) \*message)

*Creates a pdm iterator to iterate through the fields in a message.*

- LBMPDMDLL int [lbmpdm\\_iter\\_create\\_from\\_field\\_handle](#) ([lbmpdm\\_iter\\_t](#) \*\*iter, [lbmpdm\\_msg\\_t](#) \*message, [lbmpdm\\_field\\_handle\\_t](#) field\_handle)

*Creates a pdm iterator to iterate through the fields in a message starting at a particular field.*

- LBMPDMDLL int `lbmpdm_iter_delete` (`lbmpdm_iter_t` \*iter)
 

*Deletes the iterator.*
- LBMPDMDLL      `lbmpdm_field_handle_t`      `lbmpdm_iter_get_current` (`lbmpdm_iter_t` \*iter)
 

*Gets the current field handle from the iterator.*
- LBMPDMDLL int `lbmpdm_iter_first` (`lbmpdm_iter_t` \*iter)
 

*Sets the iterator back to the first field.*
- LBMPDMDLL int `lbmpdm_iter_next` (`lbmpdm_iter_t` \*iter)
 

*Steps the iterator to the next first field.*
- LBMPDMDLL uint8\_t `lbmpdm_iter_has_next` (`lbmpdm_iter_t` \*iter)
 

*Checks to see if the iterator has another field to step to.*
- LBMPDMDLL uint8\_t `lbmpdm_iter_is_current_set` (`lbmpdm_iter_t` \*iter)
 

*Checks to see if the current field is set.*
- LBMPDMDLL int `lbmpdm_iter_set_msg` (`lbmpdm_iter_t` \*iter, `lbmpdm_msg_t` \*message)
 

*Sets the message used to step through by this iterator.*
- LBMPDMDLL int `lbmpdm_iter_set_current_field_value` (`lbmpdm_iter_t` \*iter, void \*value, size\_t len)
 

*Sets the current field value to the value passed in.*
- LBMPDMDLL int `lbmpdm_iter_set_current_field_value_vec` (`lbmpdm_iter_t` \*iter, void \*value, size\_t len[], size\_t num\_arr\_elem)
 

*Sets the current field values to the passed array of values.*
- LBMPDMDLL int `lbmpdm_iter_get_current_field_value` (`lbmpdm_iter_t` \*iter, void \*value, size\_t \*len)
 

*Gets a field value from the iterator's current field.*
- LBMPDMDLL int `lbmpdm_iter_get_current_field_value_vec` (`lbmpdm_iter_t` \*iter, void \*value, size\_t len[], size\_t \*num\_arr\_elem)
 

*Gets an array of field values from the iterator's current field.*

### 8.5.1 Detailed Description

The Ultra Messaging (UM) Pre-Defined Message (PDM) API Description. Included are types, constants, and functions related to the API. Contents are subject to change.

All of the documentation and software included in this and any other Informatica Corporation Ultra Messaging Releases Copyright (C) Informatica Corporation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted only as covered by the terms of a valid software license agreement with Informatica Corporation.

Copyright (C) 2007-2014, Informatica Corporation. All Rights Reserved.

THE SOFTWARE IS PROVIDED "AS IS" AND INFORMATICA DISCLAIMS ALL WARRANTIES EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. INFORMATICA DOES NOT WARRANT THAT USE OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE. INFORMATICA SHALL NOT, UNDER ANY CIRCUMSTANCES, BE LIABLE TO LICENSEE FOR LOST PROFITS, CONSEQUENTIAL, INCIDENTAL, SPECIAL OR INDIRECT DAMAGES ARISING OUT OF OR RELATED TO THIS AGREEMENT OR THE TRANSACTIONS CONTEMPLATED HEREUNDER, EVEN IF INFORMATICA HAS BEEN APPRISED OF THE LIKELIHOOD OF SUCH DAMAGES.

The LBM Pre-Defined Message(PDM) API provides a framework for applications to create message definitions and messages from those definitions. A PDM definition contains a list of field information describing the fields that will be contained in a message. A PDM message contains one or more **fields** with each field corresponding to a specific field information object in the definition. Field info consists of:

- A name (string names or integer names are supported).
- A type (discussed below).
- If the field is required. Message fields consist of:
  - A handle to the corresponding field info.
  - A value (particular to the field type). Each named field may only appear once in a message. If multiple fields of the same name and type are needed, an array field can be used to store multiple values for that field. PDM messages can be added as a field to other PDM messages by using the field type PDM\_MESSAGE.

#### Field types

The following field types (and arrays thereof) are supported by PDM:

Description	PDM Type	C Type
Boolean	PDM_TYPE_- BOOLEAN	uint8_t
8-bit signed integer	PDM_TYPE_INT8	int8_t
8-bit unsigned integer	PDM_TYPE_UINT8	uint8_t
16-bit signed integer	PDM_TYPE_INT16	int16_t
16-bit unsigned integer	PDM_TYPE_UINT16	uint16_t
32-bit signed integer	PDM_TYPE_INT32	int32_t
32-bit unsigned integer	PDM_TYPE_UINT32	uint32_t
64-bit signed integer	PDM_TYPE_INT64	int64_t
64-bit unsigned integer	PDM_TYPE_UINT64	uint64_t
Single-precision floating point	PDM_TYPE_FLOAT	float
Double-precision floating point	PDM_TYPE_- DOUBLE	double
Decimal	PDM_TYPE_- DECIMAL	struct decimal
Timestamp	PDM_TYPE_- TIMESTAMP	struct timestamp
Fixed Length String	PDM_TYPE_FIX_- STRING	char *
String	PDM_TYPE_STRING	char *
Fixed Length Unicode	PDM_TYPE_FIX_- UNICODE	char *
Unicode	PDM_TYPE_- UNICODE	char *
Nested PDM message	PDM_TYPE_- MESSAGE	lbmpdm_msg_t *
Binary large object (BLOB)	PDM_TYPE_BLOB	void *

Note that arrays are homogeneous.

### Creating a message definition

A message definition must be defined (via [lbmpdm\\_defn\\_create\(\)](#)) before a message can be created. After creating the definition, field info can be added. Once all field information has been added to a definition, the definition must be finalized ([lbmpdm\\_defn\\_finalize\(\)](#)). Each definition must be given an id to identify that definition (and messages created with the definition) to all consumers of the message. This allows all messaging participants to define the message and for PDM to quickly deserialize the messages. As an example, we will create a simple definition with two fields, a 32-bit signed integer and a string array. We will give the definition an id of 1000.

```

lbmpdm_defn_t *defn;
lbmpdm_field_handle_t h1;
lbmpdm_field_handle_t h2;
lbmpdm_field_info_attr_t info_attr;

if (lbmpdm_defn_create(&defn, 2, 1000, 1, 0, PDM_DEFN_STR_FIELD_NAMES) != PDM_SUCCESS) {
    printf("Failed to create the definition");
    return;
}

info_attr.flags = 0;

h1 = lbmpdm_defn_add_field_info_by_str_name(defn, "Field1", PDM_TYPE_INT32, NULL);

info_attr.flags |= PDM_FIELD_INFO_FLAG_REQ;
info_attr.req = PDM_FALSE;
h2 = lbmpdm_defn_add_field_info_by_str_name(defn, "Field2", PDM_TYPE_STRING_ARR, &info_attr);

lbmpdm_defn_finalize(defn);

```

The values of the info\_attr struct will only be examined if the corresponding PDM\_FIELD\_INFO\_FLAG\_\* has been set when it is passed to the add\_field\_info function. If NULL is passed instead of an info\_attr pointer or any of the flags are not set, then default values will be used when adding the field info(which are: required = PDM\_TRUE, fixed string length = 0, and number of array elements = 0).

Once the definition exists, a message can be created and field values can be set in the message. The fields can be set by using the field handle that was returned from the call to the definition to add field info (or the field handle can be looked up from the definition).

### Sample code

Checking of return codes has been omitted but should be done during actual usage of pdm.

```

{
int rc;
lbmpdm_msg_t *msg1;
lbmpdm_msg_t *msg2;
char *msg_buffer;
int msg_len;

int32_t quant = 50;
char *str_arr[3] = {"s1", "str2", "string3"};
size_t str_len_arr[3] = {3, 5, 8};
size_t str_arr_num_elem = 3;

int32_t rcv_quant;
size_t rcv_quant_sz = sizeof(int32_t);

```

```

char *rcv_str_arr[3];
rcv_str_arr[0] = malloc(3);
rcv_str_arr[1] = malloc(5);
rcv_str_arr[2] = malloc(8);

lbmpdm_msg_create(&msg1, defn, 0);

rc = lbmpdm_msg_set_field_value(msg1, h1, &quant, 0);
rc = lbmpdm_msg_set_field_value_vec(msg1, h2, str_arr, str_len_arr, str_arr_num_elem);

msg_len = lbmpdm_msg_get_length(msg1);
msg_buffer = malloc(msg_len);
rc = lbmpdm_msg_serialize(msg1, msg_buffer);

rc = lbmpdm_msg_create(&msg2, defn, 0);
rc = lbmpdm_msg_deserialize(msg2, msg_buffer, msg_len);

rc = lbmpdm_msg_get_field_value(msg2, h1, &rcv_quant, &rcv_quant_sz);
rc = lbmpdm_msg_get_field_value_vec(msg2, h2, rcv_str_arr, str_len_arr, &str_arr_num_elem);

printf("rcv_quant = %d\n", rcv_quant);
printf("rcv_str_arr[0] = %s\n", rcv_str_arr[0]);
printf("rcv_str_arr[1] = %s\n", rcv_str_arr[1]);
printf("rcv_str_arr[2] = %s\n", rcv_str_arr[2]);

free(rcv_str_arr[0]);
free(rcv_str_arr[1]);
free(rcv_str_arr[2]);

free(msg_buffer);

lbmpdm_msg_delete(msg1);
lbmpdm_msg_delete(msg2);
lbmpdm_defn_delete(defn);
}

```

### Creating a message

Messages are created from definitions. The line above that is used to create the message:

```
lbmpdm_msg_create(&msg1, defn, 0);
```

### Setting field values in a message

Scalar (non-array) fields are added to a message via the [lbmpdm\\_msg\\_set\\_field\\_value\(\)](#) to set a field's value using its handle.

When setting a field's value, data of the appropriate type must be supplied. A 0 length can be supplied for fixed length fields but it is recommended to pass the correct size in bytes to ensure the correct number of bytes are copied into the message. As an example, to set a 32-bit signed integer field to a message:

```
rc = lbmpdm_msg_set_field_value(msg1, h1, &quant, 0);
```

Setting a string value is done by passing the char \* and the actual size of the string in bytes (normally this is the num\_chars + 1 to account for the null character but for UNICODE types this value will be larger). For the MESSAGE type, the value argument should be a lbmpdm\_msg\_t \* and the length argument should be sizeof(lbmpdm\_msg\_t \*) because the setter will attempt to access the lbmpdm\_msg\_t via the pointer and then serialize it to bytes via its serialize method.

### Setting the value of array fields in a message

To set an array's value in a message, call the [lbmpdm\\_msg\\_set\\_field\\_value\\_vec\(\)](#) API function.

As an example, the code that sets the string array field value is:

```
rc = lbmpdm_msg_set_field_value_vec(msg1, h2, str_arr, str_len_arr, str_arr_num_elem);
```

Setting an array field value requires an addition number of elements parameter and expects an array of lengths (one for each element in the array). Again, string types should pass a length array that indicates the size in bytes of each string rather than the number of characters as shown in the example code which uses sizes a length array of {3, 5, 8}.

### Serializing the message

Once a PDM message is constructed, it must be serialized for transmission. The API function [lbmpdm\\_msg\\_serialize\(\)](#) serializes the message to the buffer provided. The length of the serialized data may be obtained via the API function [lbmpdm\\_msg\\_get\\_length\(\)](#). For example, a constructed message may be sent by:

```
rc = lbmpdm_msg_serialize(msg1, msg_buffer);
rc = lbm_src_send(src, msg_buffer, msg_len, 0);
```

### Deserializing a message

When the bytes for a pdm message are received, they must be deserialized so that individual fields can be accessed. The lbmpdm\_msg\_t should be reused when possible to deserialize multiple incoming messages. This is done via the [lbmpdm\\_msg\\_deserialize\(\)](#) API function:

```
rc = lbmpdm_msg_deserialize(msg2, msg_buffer, msg_len);
// Deserializing an lbm message would be the following
// rc = lbmpdm_msg_deserialize(msg2, lbmmsg->data, lbmmsg->len);
```

### Fetching fields from a message

When fetching a field from a message, the field should be accessed by its handle.

Scalar (non-array) fields may be retrieved via the [lbmpdm\\_msg\\_get\\_field\\_value\(\)](#).

```
rc = lbmpdm_msg_get_field_value(msg2, h1, &rcv_quant, &rcv_quant_sz);
```

Array fields may be retrieved via the [lbmpdm\\_msg\\_get\\_field\\_value\\_vec\(\)](#).

```
rc = lbmpdm_msg_get_field_value_vec(msg2, h2, rcv_str_arr, str_len_arr, &str_arr_num_elem);
```

When accessing a field, it is expected that value pointer being provided already points to an area of sufficient size to hold the field's value. The len argument should indicate the size or sizes (for array types) of the available space. If the size is not sufficient, a PDM\_FAILURE will be returned and the failing len argument will be updated to indicate the actual size needed. For array types, the same logic applies to the number of elements argument, where if the number of allocated elements indicated by the input parameter is insufficient, the call will fail and the value will be updated to indicate the needed number of elements. For the MESSAGE type, the value argument should be a lbmpdm\_msg\_t \* that points to an empty lbmpdm\_msg\_t which has been created with a NULL definition. The length argument should be sizeof(lbmpdm\_msg\_t \*) because the setter will attempt to access the lbmpdm\_msg\_t via the pointer and then deserialize the bytes into it.

### Disposing of a message and definition

Once a PDM message (created by either the [lbmpdm\\_msg\\_create\(\)](#) or [lbmpdm\\_msg\\_deserialize\(\)](#) API calls) is no longer needed, it must be deleted to avoid a resource leak. This is done via the [lbmpdm\\_msg\\_delete\(\)](#) API call.

```
lbmpdm_msg_delete(msg1);
lbmpdm_msg_delete(msg2);
lbmpdm_defn_delete(defn);
```

### Error information

All functions return a value to indicate the success or failure of the operation. Most return PDM\_SUCCESS to indicate success, or PDM\_FAILURE otherwise. Consult the individual function documentation for exceptions.

The function [lbmpdm\\_errmsg\(\)](#) will return a descriptive error message.

### Additional Information

When adding arrays to a definition, an array length can be specified in the info attributes. A 0 length means that the array's length is variable and will be determined when the array is set in the actual message. A positive number for the array length will create a fixed array of that size.

When adding PDM\_TYPE\_FIX\_STRING, PDM\_TYPE\_FIX\_STRING\_ARR, PDM\_TYPE\_FIX\_UNICODE, or PDM\_TYPE\_FIX\_UNICODE\_ARR, field information to a definition, a fixed string length must be specified in the info attributes. The value should be the number of characters in the string (excluding the null character). The appropriate amount of space will be then allocated in the message for each of the expected fixed strings (for the FIX\_STRING types, there will be num\_chars + 1 bytes allocated per string and for the FIX\_UNICODE types, there will be 4 \* num\_chars + 1 bytes allocated per string). By using fixed strings for a field, as well as making the field required (and specifying a fixed size for the array types), the best performance and size can be achieved because the field will be optimized as a "fixed required" field.

When setting and getting field values of type FIX\_STRING and FIX\_UNICODE (and the corresponding arrays), extra care should be made to ensure the len parameters are correct. When setting the value, the len should indicate the actual number of bytes represented by the string that should be copied (which should include the null character). If this is less than the size indicated to the definition when setting up the field information, the rest of the space will be zeroed out. When getting the value, enough space should be allocated for the entire size of the fixed string field, which as described above should be the number of characters + 1 for the string types and 4 \* the number of characters + 1 for the unicode types.

An additional way to get a field value from a message is by using the lbmpdm\_msg\_get\_field\_value\_stct method, which does not require the storage and lengths to be allocated beforehand but instead will allocate everything during the call and set all of the appropriate values of the provided lbmpdm\_field\_value\_t. Although simpler to use, the drawback is not being able to use preallocated space to hold the field value as the other get\_field\_value methods are able to do. It also requires the application to manage the field\_value\_t and call the field\_value\_stct\_delete method when finished to clean up the allocated memory inside the field\_value\_t.

## 8.5.2 Define Documentation

### 8.5.2.1 #define PDM\_DEFN\_INT\_FIELD\_NAMES 1

PDM Field Name Type. Use integer field names.

### 8.5.2.2 #define PDM\_DEFN\_STR\_FIELD\_NAMES 0

PDM Field Name Type. Use string field names.

**8.5.2.3 #define PDM\_ERR\_CREATE\_BUFFER 11**

PDM Error Code. Error creating buffer.

**8.5.2.4 #define PDM\_ERR\_CREATE\_SECTION 10**

PDM Error Code. Error creating field section.

**8.5.2.5 #define PDM\_ERR\_DEFN\_INVALID 7**

PDM Error Code. Not a valid PDM definition.

**8.5.2.6 #define PDM\_ERR\_EINVAL 4**

PDM Error Code. Invalid parameter given.

**8.5.2.7 #define PDM\_ERR\_FIELD\_IS\_NULL 1**

PDM Error Code. Field is null.

**8.5.2.8 #define PDM\_ERR\_FIELD\_NOT\_FOUND 5**

PDM Error Code. Field not found.

**8.5.2.9 #define PDM\_ERR\_INSUFFICIENT\_BUFFER\_LENGTH 3**

PDM Error Code. Insufficient buffer length given.

**8.5.2.10 #define PDM\_ERR\_MSG\_INVALID 6**

PDM Error Code. Not a valid PDM message.

**8.5.2.11 #define PDM\_ERR\_NO\_MORE\_FIELDS 2**

PDM Error Code. No more fields to iterate over.

**8.5.2.12 #define PDM\_ERR\_NOMEM 8**

PDM Error Code. No memory available.

**8.5.2.13 #define PDM\_ERR\_REQ\_FIELD\_NOT\_SET 9**

PDM Error Code. Required field not set.

**8.5.2.14 #define PDM\_FAILURE -1**

PDM Return Code. Operation failed. See [lbmpdm\\_errnum\(\)](#) or [lbmpdm\\_errmsg\(\)](#) for the reason.

**8.5.2.15 #define PDM\_FALSE (uint8\_t) 0**

PDM true/false values. PDM value for FALSE.

**8.5.2.16 #define PDM\_FIELD\_INFO\_FLAG\_FIXED\_STR\_LEN 0x2**

PDM Field Info Flags. Field has a fixed string length (for string and unicode types).

**8.5.2.17 #define PDM\_FIELD\_INFO\_FLAG\_NUM\_ARR\_ELEM 0x4**

PDM Field Info Flags. Field has a fixed array size (for array types).

**8.5.2.18 #define PDM\_FIELD\_INFO\_FLAG\_REQ 0x1**

PDM Field Info Flags. Field is required.

**8.5.2.19 #define PDM\_INTERNAL\_TYPE\_INVALID -1**

PDM Field Type. Invalid Type.

**8.5.2.20 #define PDM\_ITER\_INVALID\_FIELD\_HANDLE -1**

PDM Iterator Invalid Handle. Indicates invalid field handle.

**8.5.2.21 #define PDM\_MSG\_FLAG\_DEL\_DEFN\_WHEN\_REPLACED 0x20**

PDM Message Flags. If a message's existing definition should be deleted when replaced when deserializing a message.

**8.5.2.22 #define PDM\_MSG\_FLAG\_INCL\_DEFN 0x2**

PDM Message Flags. If the definition should be serialized with the message.

**8.5.2.23 #define PDM\_MSG\_FLAG\_NEED\_BYTE\_SWAP 0x10**

PDM Message Flags. If the field values need bytes to be swapped (set internally).

**8.5.2.24 #define PDM\_MSG\_FLAG\_TRY\_LOAD\_DEFN\_FROM\_CACHE 0x8**

PDM Message Flags. If a message should try to load a needed definition from the cache when deserializing.

**8.5.2.25 #define PDM\_MSG\_FLAG\_USE\_MSG\_DEFN\_IF\_NEEDED 0x4**

PDM Message Flags. If a message should override its existing definition with one included when deserializing a message.

**8.5.2.26 #define PDM\_MSG\_FLAG\_VAR\_OR\_OPT\_FLDS\_SET 0x1**

PDM Message Flags. If any variable or optional fields have been set (set internally).

**8.5.2.27 #define PDM\_MSG\_VER\_POLICY\_BEST 1**

PDM Message Version Policy. Use Best Match versioning Policy.

**8.5.2.28 #define PDM\_MSG\_VER\_POLICY\_EXACT 0**

PDM Message Version Policy. Use Exact Match versioning Policy.

**8.5.2.29 #define PDM\_SUCCESS 0**

PDM Return Code. Operation was successful.

**8.5.2.30 #define PDM\_TRUE (uint8\_t) 1**

PDM true/false values. PDM value for TRUE.

**8.5.2.31 #define PDM\_TYPE\_BLOB 17**

PDM Field Type. blob (variable length).

**8.5.2.32 #define PDM\_TYPE\_BLOB\_ARR 36**

PDM Field Type. blob array.

**8.5.2.33 #define PDM\_TYPE\_BOOLEAN 0**

PDM Field Type. boolean.

**8.5.2.34 #define PDM\_TYPE\_BOOLEAN\_ARR 19**

PDM Field Type. boolean array.

**8.5.2.35 #define PDM\_TYPE\_DECIMAL 11**

PDM Field Type. decimal.

**8.5.2.36 #define PDM\_TYPE\_DECIMAL\_ARR 30**

PDM Field Type. decimal array.

**8.5.2.37 #define PDM\_TYPE\_DOUBLE 10**

PDM Field Type. double.

**8.5.2.38 #define PDM\_TYPE\_DOUBLE\_ARR 29**

PDM Field Type. double array.

**8.5.2.39 #define PDM\_TYPE\_FIX\_STRING 13**

PDM Field Type. fixed string.

**8.5.2.40 #define PDM\_TYPE\_FIX\_STRING\_ARR 32**

PDM Field Type. fixed string array.

**8.5.2.41 #define PDM\_TYPE\_FIX\_UNICODE 15**

PDM Field Type. fixed unicode.

**8.5.2.42 #define PDM\_TYPE\_FIX\_UNICODE\_ARR 34**

PDM Field Type. fixed unicode array.

**8.5.2.43 #define PDM\_TYPE\_FLOAT 9**

PDM Field Type. float.

**8.5.2.44 #define PDM\_TYPE\_FLOAT\_ARR 28**

PDM Field Type. float array.

**8.5.2.45 #define PDM\_TYPE\_INT16 3**

PDM Field Type. 16 bit integer.

**8.5.2.46 #define PDM\_TYPE\_INT16\_ARR 22**

PDM Field Type. 16 bit integer array.

**8.5.2.47 #define PDM\_TYPE\_INT32 5**

PDM Field Type. 32 bit integer.

**8.5.2.48 #define PDM\_TYPE\_INT32\_ARR 24**

PDM Field Type. 32 bit integer array.

**8.5.2.49 #define PDM\_TYPE\_INT64 7**

PDM Field Type. 64 bit integer.

**8.5.2.50 #define PDM\_TYPE\_INT64\_ARR 26**

PDM Field Type. 64 bit integer array.

**8.5.2.51 #define PDM\_TYPE\_INT8 1**

PDM Field Type. 8 bit integer.

**8.5.2.52 #define PDM\_TYPE\_INT8\_ARR 20**

PDM Field Type. 8 bit integer array.

**8.5.2.53 #define PDM\_TYPE\_MESSAGE 18**

PDM Field Type. PDM message (variable length).

**8.5.2.54 #define PDM\_TYPE\_MESSAGE\_ARR 37**

PDM Field Type. PDM message array.

**8.5.2.55 #define PDM\_TYPE\_STRING 14**

PDM Field Type. string (variable length).

**8.5.2.56 #define PDM\_TYPE\_STRING\_ARR 33**

PDM Field Type. string array.

**8.5.2.57 #define PDM\_TYPE\_TIMESTAMP 12**

PDM Field Type. timestamp.

**8.5.2.58 #define PDM\_TYPE\_TIMESTAMP\_ARR 31**

PDM Field Type. timestamp array.

**8.5.2.59 #define PDM\_TYPE\_UINT16 4**

PDM Field Type. unsigned 16 bit integer.

**8.5.2.60 #define PDM\_TYPE\_UINT16\_ARR 23**

PDM Field Type. unsigned 16 bit integer array.

**8.5.2.61 #define PDM\_TYPE\_UINT32 6**

PDM Field Type. unsigned 32 bit integer.

**8.5.2.62 #define PDM\_TYPE\_UINT32\_ARR 25**

PDM Field Type. unsigned 32 bit integer array.

**8.5.2.63 #define PDM\_TYPE\_UINT64 8**

PDM Field Type. unsigned 64 bit integer.

**8.5.2.64 #define PDM\_TYPE\_UINT64\_ARR 27**

PDM Field Type. unsigned 64 bit integer array.

**8.5.2.65 #define PDM\_TYPE\_UINT8 2**

PDM Field Type. unsigned 8 bit integer.

**8.5.2.66 #define PDM\_TYPE\_UINT8\_ARR 21**

PDM Field Type. unsigned 8 bit integer array.

**8.5.2.67 #define PDM\_TYPE\_UNICODE 16**

PDM Field Type. unicode (variable length).

**8.5.2.68 #define PDM\_TYPE\_UNICODE\_ARR 35**

PDM Field Type. unicode array.

### 8.5.3 Function Documentation

**8.5.3.1 LBMPDMExpDLL int lbmpdm\_cache\_init (uint32\_t *cache\_size*)****Parameters:**

*cache\_size* – how many buckets should this hash contain? If set to zero this will default.

**8.5.3.2 LBMPDMDLL int lbmpdm\_cache\_struct\_add (lbmpdm\_defn\_t \*  
*defn*)****Parameters:**

*defn* – the new definition structure being added.

**8.5.3.3 LBMPDMDLL int lbmpdm\_cache\_struct\_find (lbmpdm\_defn\_t \*\*  
*defn*, int32\_t *id*)****Parameters:**

*defn* – pointer to the structure to return. This field is not altered if nothing is found for a given id.

*id* – id to search for in the hash.

**8.5.3.4 LBMPDMDLL int lbmpdm\_cache\_struct\_find\_by\_version  
(lbmpdm\_defn\_t \*\* *defn*, int32\_t *id*, uint8\_t *vers\_major*, uint8\_t  
*vers\_minor*)****Parameters:**

*defn* – pointer to the structure to return. This field is not altered if nothing is found for a given id.

*id* – id to search for in the hash.

*vers\_major* – major version to search for in the hash.

*vers\_minor* – minor version to search for in the hash.

**8.5.3.5 LBMPDMDLL int lbmpdm\_cache\_struct\_remove (int32\_t *id*)****Parameters:**

*id* – id of the structure being deleted. If the id is not found this will do nothing.

**8.5.3.6 LBMPDMDLL int lbmpdm\_cache\_struct\_remove\_by\_version  
(int32\_t *id*, uint8\_t *vers\_major*, uint8\_t *vers\_minor*)****Parameters:**

*id* – id of the structure being deleted. If the id is not found this will do nothing.

*vers\_major* – major version of the definition

*vers\_minor* – minor version of the definition

**8.5.3.7 LBMPDMExpDLL lbmpdm\_field\_handle\_t lbmpdm\_defn\_add\_field\_info\_by\_int\_name (lbmpdm\_defn\_t \* *defn*, int32\_t *int\_name*, int16\_t *type*, lbmpdm\_field\_info\_attr\_t \* *info\_attr*)**

**Parameters:**

*int\_name* – the integer name  
*type* – the PDM field type  
*info\_attr* – the field information attributes

**Returns:**

PDM\_SUCCESS or PDM\_FAILURE

**8.5.3.8 LBMPDMExpDLL lbmpdm\_field\_handle\_t lbmpdm\_defn\_add\_field\_info\_by\_str\_name (lbmpdm\_defn\_t \* *defn*, const char \* *str\_name*, int16\_t *type*, lbmpdm\_field\_info\_attr\_t \* *info\_attr*)**

**Parameters:**

*str\_name* – the string name  
*type* – the PDM field type  
*info\_attr* – the field information attributes

**Returns:**

PDM\_SUCCESS or PDM\_FAILURE

**8.5.3.9 LBMPDMExpDLL int lbmpdm\_defn\_create (lbmpdm\_defn\_t \*\* *defn*, int32\_t *num\_fields*, int32\_t *id*, int8\_t *vrs\_mjr*, int8\_t *vrs\_mnr*, uint8\_t *field\_names\_type*)**

**Parameters:**

*defn* – pointer to newly created definition.  
*num\_fields* – how many fields to initially assume we will have, this is for optimization, not a limit to how many fields.  
*id* – This is the id for the definition. It is assumed that this is a unique id.  
*vrs\_mjr* – A version major number to be assigned to the newly created definition. When deserializing a message, PDM will attempt to use the existing set definition if the ids match. If the message flag is set to use the included message definition or try to load the definition from the cache, then an attempt will be made to replace the set definition with the new one by its version

numbers. Please note: when versioning message definitions, adding optional fields only is the safest way to ensure interoperability with older versions. Adding new required fields or modifying the types of existing required fields may lead to messages that are not serializable by receivers with older definition versions.

*vrs\_mnr* – A version minor number

*field\_names\_type* – A type that indicates whether the field names will be strings or ints. Use either PDM\_DEFN\_STR\_FIELD\_NAMES or PDM\_DEFN\_INT\_FIELD\_NAMES for the value.

**Returns:**

PDM\_SUCCESS or PDM\_FAILURE.

### 8.5.3.10 LBMPDMDExpDLL int lbmpdm\_defn\_delete (lbmpdm\_defn\_t \* *defn*)

**Parameters:**

*defn* – definition to be deleted.

**Returns:**

PDM\_SUCCESS or PDM\_FAILURE.

### 8.5.3.11 LBMPDMDExpDLL int lbmpdm\_defn\_deserialize (lbmpdm\_defn\_t \* *defn*, const char \* *bufptr*, uint32\_t *buflen*, uint8\_t *swap\_bytes*)

This will take the passed buffer and deserialize the contents into a newly created defn. When finished with the defn, the caller must call [lbmpdm\\_defn\\_delete\(\)](#) to properly dispose of the defn. This is done automatically by a message when the defn is included. The message normally indicates to the definition whether or not swap bytes need to be set to PDM\_TRUE so to use this method directly, this knowledge must be determined outside PDM.

**See also:**

[lbmpdm\\_msg\\_delete\(\)](#)

**Parameters:**

*defn* A pointer to a previously created PDM defn object

*bufptr* The buffer to be deserialized

*buflen* The length of the buffer.

*swap\_bytes* Whether or not bytes should be swapped to deal with endianness.

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.12 LBMPDMEExpDLL int lbmpdm\_defn\_finalize (lbmpdm\_defn\_t \* *defn*)****Parameters:**

*defn* – the definition to be finalized.

**Returns:**

PDM\_SUCCESS or PDM\_FAILURE

**8.5.3.13 LBMPDMEExpDLL lbmpdm\_field\_handle\_t lbmpdm\_defn\_get\_field\_handle\_by\_int\_name (lbmpdm\_defn\_t \* *defn*, int32\_t *int\_name*)****Parameters:**

*defn* – definition created from lbmpdm\_defn\_create.

*int\_name* – the name of the field to be retrieved.

**Returns:**

A valid field handle on success, PDM\_FAILURE otherwise.

**8.5.3.14 LBMPDMEExpDLL lbmpdm\_field\_handle\_t lbmpdm\_defn\_get\_field\_handle\_by\_str\_name (lbmpdm\_defn\_t \* *defn*, const char \* *str\_name*)****Parameters:**

*defn* – definition created from lbmpdm\_defn\_create.

*str\_name* – the name of the field to be retrieved.

**Returns:**

A valid field handle on success, PDM\_FAILURE otherwise.

**8.5.3.15 LBMPDMExpDLL int32\_t lbmpdm\_defn\_get\_field\_info\_int\_name  
(lbmpdm\_defn\_t \* *defn*, lbmpdm\_field\_handle\_t *handle*)**

**Parameters:**

*defn* – A pointer to a PDM defn object.

*handle* – A valid field handle from the definition.

**Returns:**

the int field name or -1.

**8.5.3.16 LBMPDMExpDLL const char\* lbmpdm\_defn\_get\_field\_info\_str\_name  
(lbmpdm\_defn\_t \* *defn*, lbmpdm\_field\_handle\_t  
*handle*)**

**Parameters:**

*defn* – A pointer to a PDM defn object.

*handle* – A valid field handle from the definition.

**Returns:**

the string field name or NULL.

**8.5.3.17 LBMPDMExpDLL int16\_t lbmpdm\_defn\_get\_field\_info\_type  
(lbmpdm\_defn\_t \* *defn*, lbmpdm\_field\_handle\_t *handle*)**

**Parameters:**

*defn* – A pointer to a PDM defn object.

*handle* – A valid field handle from the definition.

**Returns:**

the PDM type or PDM\_INTERNAL\_TYPE\_INVALID.

**8.5.3.18 LBMPDMExpDLL uint8\_t lbmpdm\_defn\_get\_field\_names\_type  
(lbmpdm\_defn\_t \* *defn*)**

**Parameters:**

*defn* – A pointer to a PDM defn object.

**Returns:**

the field names type

**8.5.3.19 LBMPDMDLL int32\_t lbmpdm\_defn\_get\_id (lbmpdm\_defn\_t \*  
*defn*)**

**Parameters:**

*defn* – A pointer to a PDM defn object.

**Returns:**

the id of the definition

**8.5.3.20 LBMPDMDLL uint32\_t lbmpdm\_defn\_get\_length  
(lbmpdm\_defn\_t \* *defn*)**

**Parameters:**

*defn* – A pointer to a PDM defn object.

**Returns:**

the exact length of the serialized defn

**8.5.3.21 LBMPDMDLL int8\_t lbmpdm\_defn\_get\_msg\_vers\_major  
(lbmpdm\_defn\_t \* *defn*)**

**Parameters:**

*defn* – A pointer to a PDM defn object.

**Returns:**

the major version number

**8.5.3.22 LBMPDMDLL int8\_t lbmpdm\_defn\_get\_msg\_vers\_minor  
(lbmpdm\_defn\_t \* *defn*)**

**Parameters:**

*defn* – A pointer to a PDM defn object.

**Returns:**

the minor version number

**8.5.3.23 LBMPDMExpDLL int32\_t lbmpdm\_defn\_get\_num\_fields  
(lbmpdm\_defn\_t \* *defn*)****Parameters:**

*defn* – A pointer to a PDM defn object.

**Returns:**

the number of fields in the definition

**8.5.3.24 LBMPDMExpDLL uint8\_t lbmpdm\_defn\_is\_finalized  
(lbmpdm\_defn\_t \* *defn*)****Parameters:**

*defn* – A pointer to a PDM defn object.

**Returns:**

the value indicating whether or not the definition has been finalized

**8.5.3.25 LBMPDMExpDLL int lbmpdm\_defn\_serialize (lbmpdm\_defn\_t \*  
*defn*, char \* *buffer*, uint32\_t \* *defn\_len*)****Parameters:**

*defn* A PDM defn to be serialized

*buffer* The caller allocated buffer

*defn\_len* Will be set to the length of the definition

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.26 LBMPDMExpDLL const char\* lbmpdm\_errmsg ()****Returns:**

Pointer to a static char array holding the error message.

**8.5.3.27 LBMPDMExpDLL int lbmpdm\_errnum ()****Returns:**

An integer error number.

**8.5.3.28 LBMPDMDLL int lbmpdm\_field\_value\_stct\_delete  
(lbmpdm\_field\_value\_t \**field\_value*)**

**See also:**

[lbmpdm\\_msg\\_get\\_field\\_value\\_stct](#)

**Parameters:**

*field\_value* – A pointer to a field value structure.

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.29 LBMPDMDLL int lbmpdm\_iter\_create (lbmpdm\_iter\_t \*\**iter*,  
lbmpdm\_msg\_t \**message*)**

**Parameters:**

*iter* – Pointer to the iterator pointer which will be created

*message* – the pdm message

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.30 LBMPDMDLL int lbmpdm\_iter\_create\_from\_field\_handle  
(lbmpdm\_iter\_t \*\**iter*, lbmpdm\_msg\_t \**message*,  
lbmpdm\_field\_handle\_t*field\_handle*)**

**Parameters:**

*iter* – Pointer to the iterator pointer which will be created

*message* – the pdm message

*field\_handle* – the handle to the field where the iterator should start

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.31 LBMPDMExpDLL int lbmpdm\_iter\_delete (lbmpdm\_iter\_t \* iter)****Parameters:***iter* – the pdm iterator**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.32 LBMPDMExpDLL int lbmpdm\_iter\_first (lbmpdm\_iter\_t \* iter)****Parameters:***iter* – the pdm iterator**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.33 LBMPDMExpDLL lbmpdm\_field\_handle\_t lbmpdm\_iter\_get\_current (lbmpdm\_iter\_t \* iter)****Parameters:***iter* – The pdm iterator**Returns:**

the lbmpdm\_field\_handle\_t (field handle)

**8.5.3.34 LBMPDMExpDLL int lbmpdm\_iter\_get\_current\_field\_value (lbmpdm\_iter\_t \* iter, void \* value, size\_t \* len)****Parameters:***iter* – the pdm iterator*value* – A pointer to a value big enough to hold the field value*len* – A pointer describing the currently allocated length of the void \*value. If it is not large enough to hold the field value, PDM\_FAILURE will be returned and len will be set to the needed length.*num\_arr\_elem* – The number of elements in the value and len array.**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.35 LBMPDMEExpDLL int lbmpdm\_iter\_get\_current\_field\_value\_vec  
(lbmpdm\_iter\_t \* iter, void \* value, size\_t len[ ], size\_t \* num\_arr\_elem)**

**Parameters:**

*iter* – the pdm iterator

*value* – A pointer to an array of values big enough to hold the field values

*len* – An array of lengths describing the currently allocated length of each void \*value array element. If it is not large enough to hold the field value, PDM\_FAILURE will be returned and that len element will be set to the needed length.

*num\_arr\_elem* – The number of elements in the value and len array.

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.36 LBMPDMEExpDLL uint8\_t lbmpdm\_iter\_has\_next (lbmpdm\_iter\_t \* iter)**

**Parameters:**

*iter* – the pdm iterator

**Returns:**

PDM\_TRUE if there are more fields or PDM\_FALSE if this is the last field

**8.5.3.37 LBMPDMEExpDLL uint8\_t lbmpdm\_iter\_is\_current\_set  
(lbmpdm\_iter\_t \* iter)**

**Parameters:**

*iter* – the pdm iterator

**Returns:**

PDM\_TRUE if the current field is set or PDM\_FALSE if it is not

**8.5.3.38 LBMPDMEExpDLL int lbmpdm\_iter\_next (lbmpdm\_iter\_t \* iter)**

**Parameters:**

*iter* – the pdm iterator

**Returns:**

PDM\_SUCCESS if successful or PDM\_ERR\_NO\_MORE\_FIELDS if moved beyond the last field

**8.5.3.39 LBMPDMDLL int lbmpdm\_iter\_set\_current\_field\_value  
(lbmpdm\_iter\_t \* iter, void \* value, size\_t len)****Parameters:**

*iter* – the pdm iterator  
*value* – the new field value  
*len* – the len of the enw field value

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.40 LBMPDMDLL int lbmpdm\_iter\_set\_current\_field\_value\_vec  
(lbmpdm\_iter\_t \* iter, void \* value, size\_t len[], size\_t num\_arr\_elem)****Parameters:**

*iter* – the pdm iterator  
*value* – A pointer to an array of values that should be set into the message  
*len* – A size\_t array describing the currently allocated lengths of each element in the value array For fixed length types, setting a len element to 0 will allow it to default to the fixed length size of the type.  
*num\_arr\_elem* – The number of elements in the value and len array.

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.41 LBMPDMDLL int lbmpdm\_iter\_set\_msg (lbmpdm\_iter\_t \* iter,  
lbmpdm\_msg\_t \* message)****Parameters:**

*iter* – the pdm iterator  
*message* – the pdm message to step through

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.42 LBMPDMEExpDLL int lbmpdm\_msg\_and\_defn\_delete  
(lbmpdm\_msg\_t \* *message*)**

**See also:**

[lbmpdm\\_msg\\_create\(\)](#)

**Parameters:**

*message* – A pointer to a PDM message object.

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.43 LBMPDMEExpDLL int lbmpdm\_msg\_create (lbmpdm\_msg\_t \*\*  
*message*, lbmpdm\_defn\_t \* *defn*, uint32\_t *flags*)**

**Parameters:**

*message* – pointer to the newly created message

*defn* – the definition to be used by the message

*flags* – flags to set in the message

**Returns:**

PDM\_SUCCESS or PDM\_FAILURE

**8.5.3.44 LBMPDMEExpDLL int lbmpdm\_msg\_delete (lbmpdm\_msg\_t \*  
*message*)**

**See also:**

[lbmpdm\\_msg\\_create\(\)](#)

**Parameters:**

*message* – A pointer to a PDM message object.

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.45 LBMPDMDLL int lbmpdm\_msg\_deserialize (lbmpdm\_msg\_t \*  
message, const char \* bufptr, uint32\_t buflen)**

This will take the passed buffer and deserialize the contents into a newly created message. It will verify that the buffer is a valid PDM message before trying to deserialize. When finished with the message, the caller must call [lbmpdm\\_msg\\_delete\(\)](#) to properly dispose of the message.

**See also:**

[lbmpdm\\_msg\\_delete\(\)](#)

**Parameters:**

*message* A pointer to a previously created PDM message object  
*bufptr* The buffer to be serialized  
*buflen* The length of the buffer.

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.46 LBMPDMDLL char\* lbmpdm\_msg\_get\_data (lbmpdm\_msg\_t \*  
message)****Parameters:**

*message* A PDM Message to be serialized

**Returns:**

a valid char \* or NULL if an error occurred.

**8.5.3.47 LBMPDMDLL lbmpdm\_defn\_t\* lbmpdm\_msg\_get\_defn (const  
lbmpdm\_msg\_t \* message)****Parameters:**

*message* – A pointer to a PDM message object.

**Returns:**

the message definition

---

**8.5.3.48 LBMPDMDLL int lbmpdm\_msg\_get\_field\_value (lbmpdm\_msg\_t \* message, lbmpdm\_field\_handle\_t handle, void \* value, size\_t \* len)**

**Parameters:**

*message* – A pointer to a PDM message object.

*handle* – A valid field handle

*value* – A pointer to a value big enough to hold the field value

*len* – A pointer describing the currently allocated length of the void \*value. If it is not large enough to hold the field value, PDM\_FAILURE will be returned and len will be set to the needed length.

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.49 LBMPDMDLL int lbmpdm\_msg\_get\_field\_value\_stct (lbmpdm\_msg\_t \* message, lbmpdm\_field\_handle\_t handle, lbmpdm\_field\_value\_t \* field\_value)**

**Parameters:**

*message* – A pointer to a PDM message object.

*handle* – A valid field handle

*field\_value* – A pointer to a field value structure

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.50 LBMPDMDLL int lbmpdm\_msg\_get\_field\_value\_vec (lbmpdm\_msg\_t \* message, lbmpdm\_field\_handle\_t handle, void \* value, size\_t len[ ], size\_t \* num\_arr\_elem)**

**Parameters:**

*message* – A pointer to a PDM message object.

*handle* – A valid field handle

*value* – An array of pointers with each one already allocated of sufficient length to hold each field value element

*len* – An array describing the currently allocated lengths of each element of the void \*value. If any len element is not large enough to hold the field value, PDM\_FAILURE will be returned and the len array will be set to the needed lengths.

***num\_arr\_elem*** – A pointer to the number of allocated elements in the value and len arrays. If the number is less than the number of elements in the actual field value, PDM\_FAILURE will be returned and num\_arr\_elem will be set to the correct value.

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.51 LBMPDMDExpDLL uint32\_t lbmpdm\_msg\_get\_length (const lbmpdm\_msg\_t \* message)****Parameters:**

*message* – A pointer to a PDM message object.

**Returns:**

the exact length of the serialized message

**8.5.3.52 LBMPDMDExpDLL uint8\_t lbmpdm\_msg\_is\_field\_set (lbmpdm\_msg\_t \* message, lbmpdm\_field\_handle\_t handle)****Parameters:**

*message* – A pointer to a PDM message object.

*handle* – A valid field handle

**Returns:**

PDM\_TRUE or PDM\_FALSE

**8.5.3.53 LBMPDMDExpDLL int lbmpdm\_msg\_remove\_field\_value (lbmpdm\_msg\_t \* message, lbmpdm\_field\_handle\_t handle)****Parameters:**

*message* – A pointer to a PDM message object.

*handle* – A valid field handle

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.54 LBMPDMEExpDLL int lbmpdm\_msg\_serialize (lbmpdm\_msg\_t \*  
*message*, char \* *buffer*)**

**Parameters:**

*message* A PDM Message to be serialized  
*buffer* The caller allocated buffer

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.55 LBMPDMEExpDLL int lbmpdm\_msg\_set\_field\_value (lbmpdm\_msg\_t  
\* *message*, lbmpdm\_field\_handle\_t *handle*, void \* *value*, size\_t *len*)**

**Parameters:**

*message* – A pointer to a PDM message object.  
*handle* – A valid field handle  
*value* – A pointer to a value that should be set into the message  
*len* – A size\_t describing the currently allocated length of the void \*value. For fixed length types, setting len to 0 will allow it to default to the fixed length size of the type.

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.56 LBMPDMEExpDLL int lbmpdm\_msg\_set\_field\_value\_vec  
(lbmpdm\_msg\_t \* *message*, lbmpdm\_field\_handle\_t *handle*, void \*  
*value*, size\_t *len*[ ], size\_t *num\_arr\_elem*)**

**Parameters:**

*message* – A pointer to a PDM message object.  
*handle* – A valid field handle  
*value* – A pointer to an array of values that should be set into the message  
*len* – A size\_t array describing the currently allocated lengths of each element in the value array For fixed length types, setting a len element to 0 will allow it to default to the fixed length size of the type.  
*num\_arr\_elem* – The number of elements in the value and len array.

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.57 LBMPDMExpDLL int lbmpdm\_msg\_set\_incl\_defn\_flag  
(lbmpdm\_msg\_t \* *message*)**

**Parameters:**

*message* – A pointer to a PDM message object.

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

**8.5.3.58 LBMPDMExpDLL int lbmpdm\_msg\_unset\_incl\_defn\_flag  
(lbmpdm\_msg\_t \* *message*)**

**Parameters:**

*message* – A pointer to a PDM message object.

**Returns:**

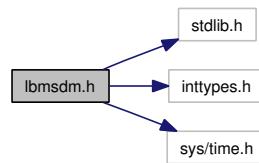
PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

## 8.6 lbmsdm.h File Reference

Ultra Messaging (UM) Self-Describing Message (SDM) API.

```
#include <stdlib.h>
#include <inttypes.h>
#include <sys/time.h>
```

Include dependency graph for lbmsdm.h:



### Data Structures

- struct `lbmsdm_decimal_t_stct`

*Structure to hold a scaled decimal number. A scaled decimal number consists of a mantissa m and an exponent exp. It represents the value  $m \cdot 10^{exp}$ .*

### Defines

- #define `LBMSDM_H_INCLUDED`
- #define `LBMSDMExpDLL`
- #define `LBMSDM_MAX_FIELD_NAME_LENGTH` 255  
*Maximum length of a field name.*
- #define `LBMSDM_TYPE_MODIFIER_ARRAY` 0x0100

### Typedefs

- typedef `lbmsdm_msg_attr_t_stct` `lbmsdm_msg_attr_t`  
*Message attributes object for SDM (opaque).*
- typedef `lbmsdm_msg_t_stct` `lbmsdm_msg_t`  
*Message object for SDM (opaque).*
- typedef `uint16_t` `lbmsdm_field_type_t`

*Type definition for an SDM field type.*

- **typedef lbmsdm\_iter\_t\_stct lbmsdm\_iter\_t**  
*Message iterator object for SDM (opaque).*
- **typedef lbmsdm\_decimal\_t\_stct lbmsdm\_decimal\_t**  
*Structure to hold a scaled decimal number. A scaled decimal number consists of a mantissa m and an exponent exp. It represents the value  $m \cdot 10^{exp}$ .*

## Enumerations

- enum {
   
**LBMSDM\_TYPE\_INVALID** = 0, **LBMSDM\_TYPE\_BOOLEAN** = 1,  
**LBMSDM\_TYPE\_INT8** = 2, **LBMSDM\_TYPE\_UINT8** = 3,  
**LBMSDM\_TYPE\_INT16** = 4, **LBMSDM\_TYPE\_UINT16** = 5, **LBMSDM\_TYPE\_INT32** = 6, **LBMSDM\_TYPE\_UINT32** = 7,  
**LBMSDM\_TYPE\_INT64** = 8, **LBMSDM\_TYPE\_UINT64** = 9, **LBMSDM\_TYPE\_FLOAT** = 10, **LBMSDM\_TYPE\_DOUBLE** = 11,  
**LBMSDM\_TYPE\_DECIMAL** = 12, **LBMSDM\_TYPE\_TIMESTAMP** = 13,  
**LBMSDM\_TYPE\_MESSAGE** = 14, **LBMSDM\_TYPE\_STRING** = 15,  
**LBMSDM\_TYPE\_UNICODE** = 16, **LBMSDM\_TYPE\_BLOB** = 17,  
**LBMSDM\_TYPE\_ARRAY\_BOOLEAN** = **LBMSDM\_TYPE\_BOOLEAN** | **LBMSDM\_TYPE\_MODIFIER\_ARRAY**, **LBMSDM\_TYPE\_ARRAY\_INT8** = **LBMSDM\_TYPE\_INT8** | **LBMSDM\_TYPE\_MODIFIER\_ARRAY**,  
**LBMSDM\_TYPE\_ARRAY\_UINT8** = (**LBMSDM\_TYPE\_UINT8** | **LBMSDM\_TYPE\_MODIFIER\_ARRAY**), **LBMSDM\_TYPE\_ARRAY\_INT16** = (**LBMSDM\_TYPE\_INT16** | **LBMSDM\_TYPE\_MODIFIER\_ARRAY**), **LBMSDM\_TYPE\_ARRAY\_UINT16** = (**LBMSDM\_TYPE\_UINT16** | **LBMSDM\_TYPE\_MODIFIER\_ARRAY**), **LBMSDM\_TYPE\_ARRAY\_INT32** = (**LBMSDM\_TYPE\_INT32** | **LBMSDM\_TYPE\_MODIFIER\_ARRAY**),  
**LBMSDM\_TYPE\_ARRAY\_UINT32** = (**LBMSDM\_TYPE\_UINT32** | **LBMSDM\_TYPE\_MODIFIER\_ARRAY**), **LBMSDM\_TYPE\_ARRAY\_INT64** = (**LBMSDM\_TYPE\_INT64** | **LBMSDM\_TYPE\_MODIFIER\_ARRAY**), **LBMSDM\_TYPE\_ARRAY\_UINT64** = (**LBMSDM\_TYPE\_UINT64** | **LBMSDM\_TYPE\_MODIFIER\_ARRAY**), **LBMSDM\_TYPE\_ARRAY\_FLOAT** = (**LBMSDM\_TYPE\_FLOAT** | **LBMSDM\_TYPE\_MODIFIER\_ARRAY**),  
**LBMSDM\_TYPE\_ARRAY\_DOUBLE** = (**LBMSDM\_TYPE\_DOUBLE** | **LBMSDM\_TYPE\_MODIFIER\_ARRAY**), **LBMSDM\_TYPE\_ARRAY\_DECIMAL** = (**LBMSDM\_TYPE\_DECIMAL** | **LBMSDM\_TYPE\_MODIFIER\_ARRAY**), **LBMSDM\_TYPE\_ARRAY\_TIMESTAMP** =

```
(LBMSDM_TYPE_TIMESTAMP | LBMSDM_TYPE_MODIFIER_ARRAY),
LBMSDM_TYPE_ARRAY_MESSAGE = (LBMSDM_TYPE_MESSAGE |  

LBMSDM_TYPE_MODIFIER_ARRAY),
LBMSDM_TYPE_ARRAY_STRING = (LBMSDM_TYPE_STRING |  

LBMSDM_TYPE_MODIFIER_ARRAY), LBMSDM_TYPE_ARRAY_  

UNICODE = (LBMSDM_TYPE_UNICODE | LBMSDM_TYPE_  

MODIFIER_ARRAY), LBMSDM_TYPE_ARRAY_BLOB = (LBMSDM_  

TYPE_BLOB | LBMSDM_TYPE_MODIFIER_ARRAY) }
```

*SDM field type definitions.*

- enum {

```
LBMSDM_SUCCESS = 0, LBMSDM_FAILURE = -1, LBMSDM_FIELD_IS_NULL = 1, LBMSDM_NO_MORE_FIELDS = 2,  

LBMSDM_INSUFFICIENT_BUFFER_LENGTH = 3 }
```

*SDM API function return codes.*

- enum {

```
LBMSDM_ERR EINVAL = 1, LBMSDM_ERR ENOMEM, LBMSDM_ERR_NAMETOOLONG, LBMSDM_ERR_DUPLICATE_FIELD,  

LBMSDM_ERR_BAD_TYPE, LBMSDM_ERR_FIELD_NOT_FOUND,  

LBMSDM_ERR_MSG_INVALID, LBMSDM_ERR_CANNOT_CONVERT,  

LBMSDM_ERR_NOT_ARRAY, LBMSDM_ERR_NOT_SCALAR,  

LBMSDM_ERR_ELEMENT_NOT_FOUND, LBMSDM_ERR_TYPE_NOT_SUPPORTED,  

LBMSDM_ERR_TYPE_MISMATCH, LBMSDM_ERR_UNICODE_CONVERSION, LBMSDM_ERR_FIELD_IS_NULL, LBMSDM_ERR_ADDING_FIELD,  

LBMSDM_ERR_ITERATOR_INVALID, LBMSDM_ERR_DELETING_FIELD, LBMSDM_ERR_INVALID_FIELD_NAME }
```

*SDM error codes.*

## Functions

- LBMSDMDLL int **lbmsdm\_errnum** (void)

*Return the error number last encountered by this thread.*

- LBMSDMDLL const char \* **lbmsdm\_errmsg** (void)

*Return an ASCII string containing the error message last encountered by this thread.*

- LBMSDMDLL int **lbmsdm\_win32\_static\_init** (void)

*Perform required initialization under Windows. This function needs to be called before any other LBM SDM API function, but only when using the static version of the LBM SDM library on Windows.*

- LBMSDMDExpDLL int [lbmsdm\\_msg\\_create](#) ([lbmsdm\\_msg\\_t](#) \*\*Message)  
*Create an SDM message to be filled in and sent.*
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_create\\_ex](#) ([lbmsdm\\_msg\\_t](#) \*\*Message, const [lbmsdm\\_attr\\_t](#) \*Attributes)  
*Create an SDM message to be filled in and sent, with options.*
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_parse](#) ([lbmsdm\\_msg\\_t](#) \*\*Message, const char \*Data, size\_t Length)  
*Create an SDM message to be parsed and processed from an existing buffer.*
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_parse\\_ex](#) ([lbmsdm\\_msg\\_t](#) \*\*Message, const char \*Data, size\_t Length, const [lbmsdm\\_attr\\_t](#) \*Attributes)  
*Create an SDM message to be parsed and processed from an existing buffer, with options.*
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_parse\\_reuse](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Data, size\_t Length)  
*Create an SDM message to be parsed and processed from an existing buffer, using an already-existing [lbmsdm\\_msg\\_t](#) structure.*
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_clone](#) ([lbmsdm\\_msg\\_t](#) \*\*Message, const [lbmsdm\\_msg\\_t](#) \*Original)  
*Clone an existing SDM message.*
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_clear](#) ([lbmsdm\\_msg\\_t](#) \*Message)  
*Clear an SDM message, deleting all fields in the message.*
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_destroy](#) ([lbmsdm\\_msg\\_t](#) \*Message)  
*Destroy an SDM message object.*
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_dump](#) ([lbmsdm\\_msg\\_t](#) \*Message, char \*Buffer, size\_t Size)  
*Dump a message into a printable string.*
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_add\\_boolean](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, uint8\_t Value)  
*Add a field to a message.*

- LBMSDMExpDLL int `lbmsdm_msg_add_int8` (`lbmsdm_msg_t` \*Message, const char \*Name, `int8_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_uint8` (`lbmsdm_msg_t` \*Message, const char \*Name, `uint8_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_int16` (`lbmsdm_msg_t` \*Message, const char \*Name, `int16_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_uint16` (`lbmsdm_msg_t` \*Message, const char \*Name, `uint16_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_int32` (`lbmsdm_msg_t` \*Message, const char \*Name, `int32_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_uint32` (`lbmsdm_msg_t` \*Message, const char \*Name, `uint32_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_int64` (`lbmsdm_msg_t` \*Message, const char \*Name, `int64_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_uint64` (`lbmsdm_msg_t` \*Message, const char \*Name, `uint64_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_float` (`lbmsdm_msg_t` \*Message, const char \*Name, float Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_double` (`lbmsdm_msg_t` \*Message, const char \*Name, double Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_decimal` (`lbmsdm_msg_t` \*Message, const char \*Name, const `lbmsdm_decimal_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_timestamp` (`lbmsdm_msg_t` \*Message, const char \*Name, const struct timeval \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_message` (`lbmsdm_msg_t` \*Message, const char \*Name, const `lbmsdm_msg_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_string` (`lbmsdm_msg_t` \*Message, const char \*Name, const char \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_unicode` (`lbmsdm_msg_t` \*Message, const char \*Name, const wchar\_t \*Value, size\_t Length)

*Add a unicode field to a message.*

- LBMSDMExpDLL int `lbmsdm_msg_add_blob` (`lbmsdm_msg_t` \*Message, const char \*Name, const void \*Value, size\_t Length)

*Add a BLOB field to a message.*

- LBMSDMExpDLL int `lbmsdm_msg_add_boolean_array` (`lbmsdm_msg_t` \*Message, const char \*Name)

*Add an array field to a message.*

- LBMSDMExpDLL int `lbmsdm_msg_add_int8_array` (`lbmsdm_msg_t` \*Message, const char \*Name)

- LBMSDMExpDLL int `lbmsdm_msg_add_uint8_array` (`lbmsdm_msg_t` \*Message, const char \*Name)

- LBMSDMDExpDLL int `lbmsdm_msg_add_int16_array` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMDExpDLL int `lbmsdm_msg_add_uint16_array` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMDExpDLL int `lbmsdm_msg_add_int32_array` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMDExpDLL int `lbmsdm_msg_add_uint32_array` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMDExpDLL int `lbmsdm_msg_add_int64_array` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMDExpDLL int `lbmsdm_msg_add_uint64_array` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMDExpDLL int `lbmsdm_msg_add_float_array` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMDExpDLL int `lbmsdm_msg_add_double_array` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMDExpDLL int `lbmsdm_msg_add_decimal_array` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMDExpDLL int `lbmsdm_msg_add_timestamp_array` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMDExpDLL int `lbmsdm_msg_add_message_array` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMDExpDLL int `lbmsdm_msg_add_string_array` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMDExpDLL int `lbmsdm_msg_add_unicode_array` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMDExpDLL int `lbmsdm_msg_add_blob_array` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMDExpDLL int `lbmsdm_msg_add_boolean_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, uint8\_t Value)

*Set the value of an array field element in a message by field index.*

- LBMSDMDExpDLL int `lbmsdm_msg_add_int8_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, int8\_t Value)
- LBMSDMDExpDLL int `lbmsdm_msg_add_uint8_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, uint8\_t Value)
- LBMSDMDExpDLL int `lbmsdm_msg_add_int16_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, int16\_t Value)
- LBMSDMDExpDLL int `lbmsdm_msg_add_uint16_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, uint16\_t Value)
- LBMSDMDExpDLL int `lbmsdm_msg_add_int32_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, int32\_t Value)
- LBMSDMDExpDLL int `lbmsdm_msg_add_uint32_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, uint32\_t Value)

- LBMSDMExpDLL int `lbmsdm_msg_add_int64_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `int64_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_uint64_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `uint64_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_float_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, float Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_double_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, double Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_decimal_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, const `lbmsdm_decimal_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_timestamp_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, const struct timeval \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_message_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, const `lbmsdm_msg_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_string_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, const char \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_unicode_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, const wchar\_t \*Value, `size_t` Length)

*Set the value of a unicode array field element in a message by field index.*

- LBMSDMExpDLL int `lbmsdm_msg_add_blob_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, const void \*Value, `size_t` Length)

*Set the value of a blob array field element in a message by field index.*

- LBMSDMExpDLL int `lbmsdm_msg_add_boolean_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `uint8_t` Value)

*Add an array field element in a message by field name.*

- LBMSDMExpDLL int `lbmsdm_msg_add_int8_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `int8_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_uint8_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `uint8_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_int16_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `int16_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_uint16_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `uint16_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_int32_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `int32_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_uint32_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `uint32_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_int64_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `int64_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_add_uint64_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `uint64_t` Value)

- LBMSDMDExpDLL int `lbmsdm_msg_add_float_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, float Value)
- LBMSDMDExpDLL int `lbmsdm_msg_add_double_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, double Value)
- LBMSDMDExpDLL int `lbmsdm_msg_add_decimal_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, const `lbmsdm_decimal_t` \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_add_timestamp_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, const struct timeval \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_add_message_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, const `lbmsdm_msg_t` \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_add_string_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, const char \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_add_unicode_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, const wchar\_t \*Value, size\_t Length)

*Add a unicode array field element in a message by field name.*

- LBMSDMDExpDLL int `lbmsdm_msg_add_blob_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, const void \*Value, size\_t Length)

*Add a BLOB array field element in a message by field name.*

- LBMSDMDExpDLL int `lbmsdm_iter_add_boolean_elem` (`lbmsdm_iter_t` \*Iterator, uint8\_t Value)

*Add an array field element in a message referenced by an iterator.*

- LBMSDMDExpDLL int `lbmsdm_iter_add_int8_elem` (`lbmsdm_iter_t` \*Iterator, int8\_t Value)
- LBMSDMDExpDLL int `lbmsdm_iter_add_uint8_elem` (`lbmsdm_iter_t` \*Iterator, uint8\_t Value)
- LBMSDMDExpDLL int `lbmsdm_iter_add_int16_elem` (`lbmsdm_iter_t` \*Iterator, int16\_t Value)
- LBMSDMDExpDLL int `lbmsdm_iter_add_uint16_elem` (`lbmsdm_iter_t` \*Iterator, uint16\_t Value)
- LBMSDMDExpDLL int `lbmsdm_iter_add_int32_elem` (`lbmsdm_iter_t` \*Iterator, int32\_t Value)
- LBMSDMDExpDLL int `lbmsdm_iter_add_uint32_elem` (`lbmsdm_iter_t` \*Iterator, uint32\_t Value)
- LBMSDMDExpDLL int `lbmsdm_iter_add_int64_elem` (`lbmsdm_iter_t` \*Iterator, int64\_t Value)
- LBMSDMDExpDLL int `lbmsdm_iter_add_uint64_elem` (`lbmsdm_iter_t` \*Iterator, uint64\_t Value)
- LBMSDMDExpDLL int `lbmsdm_iter_add_float_elem` (`lbmsdm_iter_t` \*Iterator, float Value)
- LBMSDMDExpDLL int `lbmsdm_iter_add_double_elem` (`lbmsdm_iter_t` \*Iterator, double Value)

- LBMSDMExpDLL int `lbmsdm_iter_add_decimal_elem` (`lbmsdm_iter_t` \*Iterator, const `lbmsdm_decimal_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_iter_add_timestamp_elem` (`lbmsdm_iter_t` \*Iterator, const struct timeval \*Value)
- LBMSDMExpDLL int `lbmsdm_iter_add_message_elem` (`lbmsdm_iter_t` \*Iterator, const `lbmsdm_msg_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_iter_add_string_elem` (`lbmsdm_iter_t` \*Iterator, const char \*Value)
- LBMSDMExpDLL int `lbmsdm_iter_add_unicode_elem` (`lbmsdm_iter_t` \*Iterator, const wchar\_t \*Value, size\_t Length)
 

*Add a unicode array field element in a message referenced by an iterator.*
- LBMSDMExpDLL int `lbmsdm_iter_add_blob_elem` (`lbmsdm_iter_t` \*Iterator, const void \*Value, size\_t Length)
 

*Add a BLOB array field element in a message referenced by an iterator.*
- LBMSDMExpDLL const char \* `lbmsdm_msg_get_data` (`lbmsdm_msg_t` \*Message)
 

*Get the data buffer for a constructed message, after all fields have been added to the message.*
- LBMSDMExpDLL size\_t `lbmsdm_msg_get_datalen` (`lbmsdm_msg_t` \*Message)
 

*Get the length of the data buffer for a constructed message, after all fields have been added to the message.*
- LBMSDMExpDLL int `lbmsdm_msg_get_fldcnt` (`lbmsdm_msg_t` \*Message)
 

*Get the number of fields in a message.*
- LBMSDMExpDLL int `lbmsdm_iter_create` (`lbmsdm_iter_t` \*\*Iterator, `lbmsdm_msg_t` \*Message)
 

*Create an SDM message iterator.*
- LBMSDMExpDLL int `lbmsdm_iter_destroy` (`lbmsdm_iter_t` \*Iterator)
 

*Destroy an SDM message iterator.*
- LBMSDMExpDLL int `lbmsdm_iter_first` (`lbmsdm_iter_t` \*Iterator)
 

*Position an iterator to the first field in the message.*
- LBMSDMExpDLL int `lbmsdm_iter_next` (`lbmsdm_iter_t` \*Iterator)
 

*Position an iterator to the next field in the message.*
- LBMSDMExpDLL const char \* `lbmsdm_msg_get_name_idx` (`lbmsdm_msg_t` \*Message, size\_t Index)

*Get the name of a field in a message by field index.*

- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_idx\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)

*Get the index of a field in a message by field name.*

- LBMSDMExpDLL const char \* [lbmsdm\\_iter\\_get\\_name](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)

*Get the name of the current field for an iterator.*

- LBMSDMExpDLL [lbmsdm\\_field\\_type\\_t](#) [lbmsdm\\_msg\\_get\\_type\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)

*Get the type of a field in a message by field name.*

- LBMSDMExpDLL [lbmsdm\\_field\\_type\\_t](#) [lbmsdm\\_msg\\_get\\_type\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)

*Get the type of a field in a message by field index.*

- LBMSDMExpDLL [lbmsdm\\_field\\_type\\_t](#) [lbmsdm\\_iter\\_get\\_type](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)

*Get the type of the current field for an iterator.*

- LBMSDMExpDLL int [lbmsdm\\_msg\\_is\\_null\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)

*Determine if a field in a message is null, by field name.*

- LBMSDMExpDLL int [lbmsdm\\_msg\\_is\\_null\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)

*Determine if a field in a message is null, by field index.*

- LBMSDMExpDLL int [lbmsdm\\_iter\\_is\\_null](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)

*Determine if the field referenced by an iterator is null.*

- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_elemnct\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)

*Get the number of elements in an array field in a message by field name.*

- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_element\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)

*Get the number of elements in an array field by field index.*

- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_elemnct](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)

*Get the number of elements in the current array field for an iterator.*

- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_len\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)  
*Get the length (in bytes) required for a field in a message by field name.*
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_len\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)  
*Get the length (in bytes) required for a field in a message by field index.*
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_len](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)  
*Get the length (in bytes) required for the current field for an iterator.*
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_elemlen\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, size\_t Element)  
*Get the length (in bytes) required for an array field element in a message by field name.*
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_elemlen\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, size\_t Element)  
*Get the length (in bytes) required for an array field element in a message by field index.*
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_elemlen](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element)  
*Get the length (in bytes) required for an element of the current array field for an iterator.*
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_boolean\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, uint8\_t \*Value)  
*Fetch a field value from a message by field index.*
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_int8\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, int8\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_uint8\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, uint8\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_int16\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, int16\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_uint16\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, uint16\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_int32\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, int32\_t \*Value)
- LBMSDMExpDLL int [lbmsdm\\_msg\\_get\\_uint32\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, uint32\_t \*Value)

- LBMSDMDExpDLL int `lbmsdm_msg_get_int64_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, int64\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_uint64_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, uint64\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_float_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, float \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_double_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, double \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_decimal_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, `lbmsdm_decimal_t` \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_timestamp_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, struct timeval \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_message_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, `lbmsdm_msg_t` \*\*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_string_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, char \*Value, size\_t \*Size)

*Fetch a string field value from a message by field index.*
- LBMSDMDExpDLL int `lbmsdm_msg_get_unicode_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, wchar\_t \*Value, size\_t \*Size)

*Fetch a unicode field value from a message by field index.*
- LBMSDMDExpDLL int `lbmsdm_msg_get_blob_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, void \*Value, size\_t \*Size)

*Fetch a BLOB field value from a message by field index.*
- LBMSDMDExpDLL int `lbmsdm_msg_get_boolean_name` (`lbmsdm_msg_t` \*Message, const char \*Name, uint8\_t \*Value)

*Fetch a field value from a message by field name.*
- LBMSDMDExpDLL int `lbmsdm_msg_get_int8_name` (`lbmsdm_msg_t` \*Message, const char \*Name, int8\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_uint8_name` (`lbmsdm_msg_t` \*Message, const char \*Name, uint8\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_int16_name` (`lbmsdm_msg_t` \*Message, const char \*Name, int16\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_uint16_name` (`lbmsdm_msg_t` \*Message, const char \*Name, uint16\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_int32_name` (`lbmsdm_msg_t` \*Message, const char \*Name, int32\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_uint32_name` (`lbmsdm_msg_t` \*Message, const char \*Name, uint32\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_int64_name` (`lbmsdm_msg_t` \*Message, const char \*Name, int64\_t \*Value)

- LBMSDMExpDLL int `lbmsdm_msg_get_uint64_name` (`lbmsdm_msg_t` \*Message, const char \*Name, uint64\_t \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_float_name` (`lbmsdm_msg_t` \*Message, const char \*Name, float \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_double_name` (`lbmsdm_msg_t` \*Message, const char \*Name, double \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_decimal_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `lbmsdm_decimal_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_timestamp_name` (`lbmsdm_msg_t` \*Message, const char \*Name, struct timeval \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_message_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `lbmsdm_msg_t` \*\*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_string_name` (`lbmsdm_msg_t` \*Message, const char \*Name, char \*Value, size\_t \*Size)
 

*Fetch a string field value from a message by field name.*
- LBMSDMExpDLL int `lbmsdm_msg_get_unicode_name` (`lbmsdm_msg_t` \*Message, const char \*Name, wchar\_t \*Value, size\_t \*Size)
 

*Fetch a unicode field value from a message by field name.*
- LBMSDMExpDLL int `lbmsdm_msg_get_blob_name` (`lbmsdm_msg_t` \*Message, const char \*Name, void \*Value, size\_t \*Size)
 

*Fetch a BLOB field value from a message by field name.*
- LBMSDMExpDLL int `lbmsdm_iter_get_boolean` (`lbmsdm_iter_t` \*Iterator, uint8\_t \*Value)
 

*Fetch a field value from the field referenced by an iterator.*
- LBMSDMExpDLL int `lbmsdm_iter_get_int8` (`lbmsdm_iter_t` \*Iterator, int8\_t \*Value)
- LBMSDMExpDLL int `lbmsdm_iter_get_uint8` (`lbmsdm_iter_t` \*Iterator, uint8\_t \*Value)
- LBMSDMExpDLL int `lbmsdm_iter_get_int16` (`lbmsdm_iter_t` \*Iterator, int16\_t \*Value)
- LBMSDMExpDLL int `lbmsdm_iter_get_uint16` (`lbmsdm_iter_t` \*Iterator, uint16\_t \*Value)
- LBMSDMExpDLL int `lbmsdm_iter_get_int32` (`lbmsdm_iter_t` \*Iterator, int32\_t \*Value)
- LBMSDMExpDLL int `lbmsdm_iter_get_uint32` (`lbmsdm_iter_t` \*Iterator, uint32\_t \*Value)
- LBMSDMExpDLL int `lbmsdm_iter_get_int64` (`lbmsdm_iter_t` \*Iterator, int64\_t \*Value)
- LBMSDMExpDLL int `lbmsdm_iter_get_uint64` (`lbmsdm_iter_t` \*Iterator, uint64\_t \*Value)

- LBMSDMDExpDLL int `lbmsdm_iter_get_float` (`lbmsdm_iter_t` \*Iterator, float \*Value)
- LBMSDMDExpDLL int `lbmsdm_iter_get_double` (`lbmsdm_iter_t` \*Iterator, double \*Value)
- LBMSDMDExpDLL int `lbmsdm_iter_get_decimal` (`lbmsdm_iter_t` \*Iterator, `lbmsdm_decimal_t` \*Value)
- LBMSDMDExpDLL int `lbmsdm_iter_get_timestamp` (`lbmsdm_iter_t` \*Iterator, struct timeval \*Value)
- LBMSDMDExpDLL int `lbmsdm_iter_get_message` (`lbmsdm_iter_t` \*Iterator, `lbmsdm_msg_t` \*\*Value)
- LBMSDMDExpDLL int `lbmsdm_iter_get_string` (`lbmsdm_iter_t` \*Iterator, char \*Value, size\_t \*Size)

*Fetch a string field value from the field referenced by an iterator.*

- LBMSDMDExpDLL int `lbmsdm_iter_get_unicode` (`lbmsdm_iter_t` \*Iterator, wchar\_t \*Value, size\_t \*Size)

*Fetch a unicode field value from the field referenced by an iterator.*

- LBMSDMDExpDLL int `lbmsdm_iter_get_blob` (`lbmsdm_iter_t` \*Iterator, void \*Value, size\_t \*Size)

*Fetch a BLOB field value from the field referenced by an iterator.*

- LBMSDMDExpDLL int `lbmsdm_msg_get_boolean_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, uint8\_t \*Value)

*Fetch an array field element value from a message by field index.*

- LBMSDMDExpDLL int `lbmsdm_msg_get_int8_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, int8\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_uint8_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, uint8\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_int16_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, int16\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_uint16_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, uint16\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_int32_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, int32\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_uint32_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, uint32\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_int64_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, int64\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_uint64_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, uint64\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_float_elem_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, size\_t Element, float \*Value)

- LBMSDMExpDLL int `lbmsdm_msg_get_double_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `size_t` Element, `double` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_decimal_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `size_t` Element, `lbmsdm_decimal_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_timestamp_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `size_t` Element, `struct timeval` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_message_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `size_t` Element, `lbmsdm_msg_t` \*\*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_string_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `size_t` Element, `char` \*Value, `size_t` \*Size)
 

*Fetch a string array field element value from a message by field index.*
- LBMSDMExpDLL int `lbmsdm_msg_get_unicode_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `size_t` Element, `wchar_t` \*Value, `size_t` \*Size)
 

*Fetch a unicode array field element value from a message by field index.*
- LBMSDMExpDLL int `lbmsdm_msg_get_blob_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `size_t` Element, `void` \*Value, `size_t` \*Size)
 

*Fetch a BLOB array field element value from a message by field index.*
- LBMSDMExpDLL int `lbmsdm_msg_get_boolean_elem_name` (`lbmsdm_msg_t` \*Message, `const char` \*Name, `size_t` Element, `uint8_t` \*Value)
 

*Fetch an array field element value from a message by field name.*
- LBMSDMExpDLL int `lbmsdm_msg_get_int8_elem_name` (`lbmsdm_msg_t` \*Message, `const char` \*Name, `size_t` Element, `int8_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_uint8_elem_name` (`lbmsdm_msg_t` \*Message, `const char` \*Name, `size_t` Element, `uint8_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_int16_elem_name` (`lbmsdm_msg_t` \*Message, `const char` \*Name, `size_t` Element, `int16_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_uint16_elem_name` (`lbmsdm_msg_t` \*Message, `const char` \*Name, `size_t` Element, `uint16_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_int32_elem_name` (`lbmsdm_msg_t` \*Message, `const char` \*Name, `size_t` Element, `int32_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_uint32_elem_name` (`lbmsdm_msg_t` \*Message, `const char` \*Name, `size_t` Element, `uint32_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_int64_elem_name` (`lbmsdm_msg_t` \*Message, `const char` \*Name, `size_t` Element, `int64_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_uint64_elem_name` (`lbmsdm_msg_t` \*Message, `const char` \*Name, `size_t` Element, `uint64_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_float_elem_name` (`lbmsdm_msg_t` \*Message, `const char` \*Name, `size_t` Element, `float` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_get_double_elem_name` (`lbmsdm_msg_t` \*Message, `const char` \*Name, `size_t` Element, `double` \*Value)

- LBMSDMDExpDLL int `lbmsdm_msg_get_decimal_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, size\_t Element, `lbmsdm_decimal_t` \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_timestamp_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, size\_t Element, struct timeval \*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_message_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, size\_t Element, `lbmsdm_msg_t` \*\*Value)
- LBMSDMDExpDLL int `lbmsdm_msg_get_string_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, size\_t Element, char \*Value, size\_t \*Size)

*Fetch a string array field element value from a message by field name.*
- LBMSDMDExpDLL int `lbmsdm_msg_get_unicode_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, size\_t Element, wchar\_t \*Value, size\_t \*Size)

*Fetch a unicode array field element value from a message by field name.*
- LBMSDMDExpDLL int `lbmsdm_msg_get_blob_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, size\_t Element, void \*Value, size\_t \*Size)

*Fetch a BLOB array field element value from a message by field name.*
- LBMSDMDExpDLL int `lbmsdm_iter_get_boolean_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, uint8\_t \*Value)

*Fetch an array field element value from the field referenced by an iterator.*
- LBMSDMDExpDLL int `lbmsdm_iter_get_int8_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, int8\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_iter_get_uint8_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, uint8\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_iter_get_int16_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, int16\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_iter_get_uint16_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, uint16\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_iter_get_int32_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, int32\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_iter_get_uint32_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, uint32\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_iter_get_int64_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, int64\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_iter_get_uint64_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, uint64\_t \*Value)
- LBMSDMDExpDLL int `lbmsdm_iter_get_float_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, float \*Value)
- LBMSDMDExpDLL int `lbmsdm_iter_get_double_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, double \*Value)
- LBMSDMDExpDLL int `lbmsdm_iter_get_decimal_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, `lbmsdm_decimal_t` \*Value)

- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_timestamp\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, struct timeval \*Value)
  - LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_message\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, [lbmsdm\\_msg\\_t](#) \*\*Value)
  - LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_string\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, char \*Value, size\_t \*Size)
 

*Fetch a string array field element value from the field referenced by an iterator.*
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_unicode\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, wchar\_t \*Value, size\_t \*Size)
 

*Fetch a unicode array field element value from the field referenced by an iterator.*
- LBMSDMExpDLL int [lbmsdm\\_iter\\_get\\_blob\\_elem](#) ([lbmsdm\\_iter\\_t](#) \*Iterator, size\_t Element, void \*Value, size\_t \*Size)
 

*Fetch a blob array field element value from the field referenced by an iterator.*
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_null\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name)
 

*Set a field in a message to null, by field name.*
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_null\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index)
 

*Set a field in a message to null, by field index.*
- LBMSDMExpDLL int [lbmsdm\\_iter\\_set\\_null](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)
 

*Set the field referenced by an iterator to null.*
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_boolean\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, uint8\_t Value)
 

*Set a field value in a message by field index.*
- LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_int8\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, int8\_t Value)
  - LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_uint8\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, uint8\_t Value)
  - LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_int16\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, int16\_t Value)
  - LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_uint16\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, uint16\_t Value)
  - LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_int32\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, int32\_t Value)
  - LBMSDMExpDLL int [lbmsdm\\_msg\\_set\\_uint32\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, uint32\_t Value)

- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_int64\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, int64\_t Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_uint64\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, uint64\_t Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_float\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, float Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_double\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, double Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_decimal\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, const [lbmsdm\\_decimal\\_t](#) \*Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_timestamp\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, const struct timeval \*Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_message\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, const [lbmsdm\\_msg\\_t](#) \*Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_string\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, const char \*Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_unicode\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, const wchar\_t \*Value, size\_t Length)

*Set a unicode field value in a message by field index.*

- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_blob\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, const void \*Value, size\_t Length)

*Set a BLOB field value in a message by field index.*

- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_boolean\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, uint8\_t Value)

*Set a field value in a message by field name.*

- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_int8\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, int8\_t Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_uint8\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, uint8\_t Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_int16\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, int16\_t Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_uint16\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, uint16\_t Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_int32\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, int32\_t Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_uint32\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, uint32\_t Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_int64\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, int64\_t Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_uint64\\_name](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, uint64\_t Value)

- LBMSDMExpDLL int `lbmsdm_msg_set_float_name` (`lbmsdm_msg_t` \*Message, const char \*Name, float Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_double_name` (`lbmsdm_msg_t` \*Message, const char \*Name, double Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_decimal_name` (`lbmsdm_msg_t` \*Message, const char \*Name, const `lbmsdm_decimal_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_timestamp_name` (`lbmsdm_msg_t` \*Message, const char \*Name, const struct timeval \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_message_name` (`lbmsdm_msg_t` \*Message, const char \*Name, const `lbmsdm_msg_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_string_name` (`lbmsdm_msg_t` \*Message, const char \*Name, const char \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_unicode_name` (`lbmsdm_msg_t` \*Message, const char \*Name, const wchar\_t \*Value, size\_t Length)
 

*Set a unicode field value in a message by field name.*
- LBMSDMExpDLL int `lbmsdm_msg_set_blob_name` (`lbmsdm_msg_t` \*Message, const char \*Name, const void \*Value, size\_t Length)
 

*Set a BLOB field value in a message by field name.*
- LBMSDMExpDLL int `lbmsdm_iter_set_boolean` (`lbmsdm_iter_t` \*Iterator, uint8\_t Value)
 

*Set a field value in the field referenced by an iterator.*
- LBMSDMExpDLL int `lbmsdm_iter_set_int8` (`lbmsdm_iter_t` \*Iterator, int8\_t Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_uint8` (`lbmsdm_iter_t` \*Iterator, uint8\_t Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_int16` (`lbmsdm_iter_t` \*Iterator, int16\_t Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_uint16` (`lbmsdm_iter_t` \*Iterator, uint16\_t Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_int32` (`lbmsdm_iter_t` \*Iterator, int32\_t Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_uint32` (`lbmsdm_iter_t` \*Iterator, uint32\_t Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_int64` (`lbmsdm_iter_t` \*Iterator, int64\_t Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_uint64` (`lbmsdm_iter_t` \*Iterator, uint64\_t Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_float` (`lbmsdm_iter_t` \*Iterator, float Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_double` (`lbmsdm_iter_t` \*Iterator, double Value)

- LBMSDMDExpDLL int `lbmsdm_iter_set_decimal` (`lbmsdm_iter_t` \*Iterator, const `lbmsdm_decimal_t` \*Value)
- LBMSDMDExpDLL int `lbmsdm_iter_set_timestamp` (`lbmsdm_iter_t` \*Iterator, const struct timeval \*Value)
- LBMSDMDExpDLL int `lbmsdm_iter_set_message` (`lbmsdm_iter_t` \*Iterator, const `lbmsdm_msg_t` \*Value)
- LBMSDMDExpDLL int `lbmsdm_iter_set_string` (`lbmsdm_iter_t` \*Iterator, const char \*Value)
- LBMSDMDExpDLL int `lbmsdm_iter_set_unicode` (`lbmsdm_iter_t` \*Iterator, const wchar\_t \*Value, size\_t Length)

*Set a unicode field value in the field referenced by an iterator.*
- LBMSDMDExpDLL int `lbmsdm_iter_set_blob` (`lbmsdm_iter_t` \*Iterator, const void \*Value, size\_t Length)

*Set a BLOB field value in the field referenced by an iterator.*
- LBMSDMDExpDLL int `lbmsdm_msg_set_boolean_array_idx` (`lbmsdm_msg_t` \*Message, size\_t Index)

*Set a field in a message by field index to an array field.*

  - LBMSDMDExpDLL int `lbmsdm_msg_set_int8_array_idx` (`lbmsdm_msg_t` \*Message, size\_t Index)
  - LBMSDMDExpDLL int `lbmsdm_msg_set_uint8_array_idx` (`lbmsdm_msg_t` \*Message, size\_t Index)
  - LBMSDMDExpDLL int `lbmsdm_msg_set_int16_array_idx` (`lbmsdm_msg_t` \*Message, size\_t Index)
  - LBMSDMDExpDLL int `lbmsdm_msg_set_uint16_array_idx` (`lbmsdm_msg_t` \*Message, size\_t Index)
  - LBMSDMDExpDLL int `lbmsdm_msg_set_int32_array_idx` (`lbmsdm_msg_t` \*Message, size\_t Index)
  - LBMSDMDExpDLL int `lbmsdm_msg_set_uint32_array_idx` (`lbmsdm_msg_t` \*Message, size\_t Index)
  - LBMSDMDExpDLL int `lbmsdm_msg_set_int64_array_idx` (`lbmsdm_msg_t` \*Message, size\_t Index)
  - LBMSDMDExpDLL int `lbmsdm_msg_set_uint64_array_idx` (`lbmsdm_msg_t` \*Message, size\_t Index)
  - LBMSDMDExpDLL int `lbmsdm_msg_set_float_array_idx` (`lbmsdm_msg_t` \*Message, size\_t Index)
  - LBMSDMDExpDLL int `lbmsdm_msg_set_double_array_idx` (`lbmsdm_msg_t` \*Message, size\_t Index)
  - LBMSDMDExpDLL int `lbmsdm_msg_set_decimal_array_idx` (`lbmsdm_msg_t` \*Message, size\_t Index)
  - LBMSDMDExpDLL int `lbmsdm_msg_set_timestamp_array_idx` (`lbmsdm_msg_t` \*Message, size\_t Index)

- LBMSDMExpDLL int `lbmsdm_msg_set_message_array_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index)
- LBMSDMExpDLL int `lbmsdm_msg_set_string_array_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index)
- LBMSDMExpDLL int `lbmsdm_msg_set_unicode_array_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index)
- LBMSDMExpDLL int `lbmsdm_msg_set_blob_array_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index)
- LBMSDMExpDLL int `lbmsdm_msg_set_boolean_array_name` (`lbmsdm_msg_t` \*Message, const char \*Name)

*Set a field in a message by field name to an array field.*
- LBMSDMExpDLL int `lbmsdm_msg_set_int8_array_name` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMExpDLL int `lbmsdm_msg_set_uint8_array_name` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMExpDLL int `lbmsdm_msg_set_int16_array_name` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMExpDLL int `lbmsdm_msg_set_uint16_array_name` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMExpDLL int `lbmsdm_msg_set_int32_array_name` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMExpDLL int `lbmsdm_msg_set_uint32_array_name` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMExpDLL int `lbmsdm_msg_set_int64_array_name` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMExpDLL int `lbmsdm_msg_set_uint64_array_name` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMExpDLL int `lbmsdm_msg_set_float_array_name` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMExpDLL int `lbmsdm_msg_set_double_array_name` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMExpDLL int `lbmsdm_msg_set_decimal_array_name` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMExpDLL int `lbmsdm_msg_set_timestamp_array_name` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMExpDLL int `lbmsdm_msg_set_message_array_name` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMExpDLL int `lbmsdm_msg_set_string_array_name` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMExpDLL int `lbmsdm_msg_set_unicode_array_name` (`lbmsdm_msg_t` \*Message, const char \*Name)
- LBMSDMExpDLL int `lbmsdm_msg_set_blob_array_name` (`lbmsdm_msg_t` \*Message, const char \*Name)

- LBMSDMDExpDLL int [lbmsdm\\_iter\\_set\\_boolean\\_array](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)

*Set a field in a message by field name to an array field.*

- LBMSDMDExpDLL int [lbmsdm\\_iter\\_set\\_int8\\_array](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)
- LBMSDMDExpDLL int [lbmsdm\\_iter\\_set\\_uint8\\_array](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)
- LBMSDMDExpDLL int [lbmsdm\\_iter\\_set\\_int16\\_array](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)
- LBMSDMDExpDLL int [lbmsdm\\_iter\\_set\\_uint16\\_array](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)
- LBMSDMDExpDLL int [lbmsdm\\_iter\\_set\\_int32\\_array](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)
- LBMSDMDExpDLL int [lbmsdm\\_iter\\_set\\_uint32\\_array](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)
- LBMSDMDExpDLL int [lbmsdm\\_iter\\_set\\_int64\\_array](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)
- LBMSDMDExpDLL int [lbmsdm\\_iter\\_set\\_uint64\\_array](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)
- LBMSDMDExpDLL int [lbmsdm\\_iter\\_set\\_float\\_array](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)
- LBMSDMDExpDLL int [lbmsdm\\_iter\\_set\\_double\\_array](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)
- LBMSDMDExpDLL int [lbmsdm\\_iter\\_set\\_decimal\\_array](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)
- LBMSDMDExpDLL int [lbmsdm\\_iter\\_set\\_timestamp\\_array](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)
- LBMSDMDExpDLL int [lbmsdm\\_iter\\_set\\_message\\_array](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)
- LBMSDMDExpDLL int [lbmsdm\\_iter\\_set\\_string\\_array](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)
- LBMSDMDExpDLL int [lbmsdm\\_iter\\_set\\_unicode\\_array](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)
- LBMSDMDExpDLL int [lbmsdm\\_iter\\_set\\_blob\\_array](#) ([lbmsdm\\_iter\\_t](#) \*Iterator)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_boolean\\_elem\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, size\_t Element, uint8\_t Value)

*Set the value of an array field element in a message by field index.*

- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_int8\\_elem\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, size\_t Element, int8\_t Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_uint8\\_elem\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, size\_t Element, uint8\_t Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_int16\\_elem\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, size\_t Element, int16\_t Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_uint16\\_elem\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, size\_t Element, uint16\_t Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_int32\\_elem\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, size\_t Element, int32\_t Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_uint32\\_elem\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, size\_t Element, uint32\_t Value)
- LBMSDMDExpDLL int [lbmsdm\\_msg\\_set\\_int64\\_elem\\_idx](#) ([lbmsdm\\_msg\\_t](#) \*Message, size\_t Index, size\_t Element, int64\_t Value)

- LBMSDMExpDLL int `lbmsdm_msg_set_uint64_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `size_t` Element, `uint64_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_float_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `size_t` Element, `float` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_double_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `size_t` Element, `double` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_decimal_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `size_t` Element, const `lbmsdm_decimal_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_timestamp_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `size_t` Element, const struct `timeval` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_message_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `size_t` Element, const `lbmsdm_msg_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_string_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `size_t` Element, const `char` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_unicode_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `size_t` Element, const `wchar_t` \*Value, `size_t` Length)
 

*Set the value of a unicode array field element in a message by field index.*
- LBMSDMExpDLL int `lbmsdm_msg_set_blob_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `size_t` Element, const `void` \*Value, `size_t` Length)
 

*Set the value of a BLOB array field element in a message by field index.*
- LBMSDMExpDLL int `lbmsdm_msg_set_boolean_elem_name` (`lbmsdm_msg_t` \*Message, const `char` \*Name, `size_t` Element, `uint8_t` Value)
 

*Set the value of an array field element in a message by field name.*
- LBMSDMExpDLL int `lbmsdm_msg_set_int8_elem_name` (`lbmsdm_msg_t` \*Message, const `char` \*Name, `size_t` Element, `int8_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_uint8_elem_name` (`lbmsdm_msg_t` \*Message, const `char` \*Name, `size_t` Element, `uint8_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_int16_elem_name` (`lbmsdm_msg_t` \*Message, const `char` \*Name, `size_t` Element, `int16_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_uint16_elem_name` (`lbmsdm_msg_t` \*Message, const `char` \*Name, `size_t` Element, `uint16_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_int32_elem_name` (`lbmsdm_msg_t` \*Message, const `char` \*Name, `size_t` Element, `int32_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_uint32_elem_name` (`lbmsdm_msg_t` \*Message, const `char` \*Name, `size_t` Element, `uint32_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_int64_elem_name` (`lbmsdm_msg_t` \*Message, const `char` \*Name, `size_t` Element, `int64_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_uint64_elem_name` (`lbmsdm_msg_t` \*Message, const `char` \*Name, `size_t` Element, `uint64_t` Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_float_elem_name` (`lbmsdm_msg_t` \*Message, const `char` \*Name, `size_t` Element, `float` Value)

- LBMSDMExpDLL int `lbmsdm_msg_set_double_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, size\_t Element, double Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_decimal_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, size\_t Element, const `lbmsdm_decimal_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_timestamp_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, size\_t Element, const struct timeval \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_message_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, size\_t Element, const `lbmsdm_msg_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_string_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, size\_t Element, const char \*Value)
- LBMSDMExpDLL int `lbmsdm_msg_set_unicode_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, size\_t Element, const wchar\_t \*Value, size\_t Length)

*Set the value of a unicode array field element in a message by field name.*

- LBMSDMExpDLL int `lbmsdm_msg_set_blob_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, size\_t Element, const void \*Value, size\_t Length)

*Set the value of a BLOB array field element in a message by field name.*

- LBMSDMExpDLL int `lbmsdm_iter_set_boolean_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, uint8\_t Value)

*Set the value of an array field element in the field referenced by an iterator.*

- LBMSDMExpDLL int `lbmsdm_iter_set_int8_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, int8\_t Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_uint8_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, uint8\_t Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_int16_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, int16\_t Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_uint16_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, uint16\_t Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_int32_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, int32\_t Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_uint32_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, uint32\_t Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_int64_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, int64\_t Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_uint64_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, uint64\_t Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_float_elem` (`lbmsdm_iter_t` \*Iterator, size\_t Element, float Value)

- LBMSDMExpDLL int `lbmsdm_iter_set_double_elem` (`lbmsdm_iter_t` \*Iterator, `size_t` Element, double Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_decimal_elem` (`lbmsdm_iter_t` \*Iterator, `size_t` Element, const `lbmsdm_decimal_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_timestamp_elem` (`lbmsdm_iter_t` \*Iterator, `size_t` Element, const struct timeval \*Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_message_elem` (`lbmsdm_iter_t` \*Iterator, `size_t` Element, const `lbmsdm_msg_t` \*Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_string_elem` (`lbmsdm_iter_t` \*Iterator, `size_t` Element, const char \*Value)
- LBMSDMExpDLL int `lbmsdm_iter_set_unicode_elem` (`lbmsdm_iter_t` \*Iterator, `size_t` Element, const wchar\_t \*Value, `size_t` Length)

*Set the value of a unicode array field element in the field referenced by an iterator.*

- LBMSDMExpDLL int `lbmsdm_iter_set_blob_elem` (`lbmsdm_iter_t` \*Iterator, `size_t` Element, const void \*Value, `size_t` Length)

*Set the value of a BLOB array field element in the field referenced by an iterator.*

- LBMSDMExpDLL int `lbmsdm_msg_del_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index)

*Delete a field from a message by field index.*

- LBMSDMExpDLL int `lbmsdm_msg_del_name` (`lbmsdm_msg_t` \*Message, const char \*Name)

*Delete a field from a message by field name.*

- LBMSDMExpDLL int `lbmsdm_iter_del` (`lbmsdm_iter_t` \*Iterator)

*Delete a field referenced by an iterator.*

- LBMSDMExpDLL int `lbmsdm_msg_del_elem_idx` (`lbmsdm_msg_t` \*Message, `size_t` Index, `size_t` Element)

*Delete an element from an array field by field index.*

- LBMSDMExpDLL int `lbmsdm_msg_del_elem_name` (`lbmsdm_msg_t` \*Message, const char \*Name, `size_t` Element)

*Delete an element from an array field by field name.*

- LBMSDMExpDLL int `lbmsdm_iter_del_elem` (`lbmsdm_iter_t` \*Iterator, `size_t` Element)

*Delete an element from an array field referenced by an iterator.*

- LBMSDMExpDLL int `lbmsdm_msg_attr_create` (`lbmsdm_msg_attr_t` \*\*Attributes)

*Create and fill an SDM message attribute object with the default values.*

- LBMSDMExpDLL int `lbmsdm_msg_attr_delete` (`lbmsdm_msg_attr_t *Attributes`)

*Delete an SDM message attribute object.*

- LBMSDMExpDLL int `lbmsdm_msg_attr_dup` (`lbmsdm_msg_attr_t **Attributes, lbmsdm_msg_attr_t *Original`)

*Duplicate an SDM message attribute object.*

- LBMSDMExpDLL int `lbmsdm_msg_attr_setopt` (`lbmsdm_msg_attr_t *Attributes, const char *Option, void *Value, size_t Length`)

*Set an option for the given SDM message attribute object.*

- LBMSDMExpDLL int `lbmsdm_msg_attr_str_setopt` (`lbmsdm_msg_attr_t *Attributes, const char *Option, const char *Value)`

*Set an option for the given SDM message attribute object using a string.*

- LBMSDMExpDLL int `lbmsdm_msg_attr_getopt` (`lbmsdm_msg_attr_t *Attributes, const char *Option, void *Value, size_t *Length`)

*Retrieve the value of an option for the given SDM message attribute.*

- LBMSDMExpDLL int `lbmsdm_msg_attr_str_getopt` (`lbmsdm_msg_attr_t *Attributes, const char *Option, char *Value, size_t *Length`)

*Retrieve the value of an option for the given SDM message attribute as a string.*

### 8.6.1 Detailed Description

#### Author:

David K. Ameiss - Informatica Corporation

#### VersIdn:

//UMprod/REL\_5\_3\_6/29West/lbm/src/sdm/lbm/lbmsdm.h#2

The Ultra Messaging (UM) Self-Describing Message (SDM) API Description. Included are types, constants, and functions related to the API. Contents are subject to change.

All of the documentation and software included in this and any other Informatica Corporation Ultra Messaging Releases Copyright (C) Informatica Corporation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted only as covered by the terms of a valid software license agreement with Informatica Corporation.

Copyright (C) 2007-2014, Informatica Corporation. All Rights Reserved.

THE SOFTWARE IS PROVIDED "AS IS" AND INFORMATICA DISCLAIMS ALL WARRANTIES EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. INFORMATICA DOES NOT WARRANT THAT USE OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE. INFORMATICA SHALL NOT, UNDER ANY CIRCUMSTANCES, BE LIABLE TO LICENSEE FOR LOST PROFITS, CONSEQUENTIAL, INCIDENTAL, SPECIAL OR INDIRECT DAMAGES ARISING OUT OF OR RELATED TO THIS AGREEMENT OR THE TRANSACTIONS CONTEMPLATED HEREUNDER, EVEN IF INFORMATICA HAS BEEN APPRISED OF THE LIKELIHOOD OF SUCH DAMAGES.

The LBM Self-Describing Message (SDM) API provides a framework for applications to create and use messages containing self-describing data (name and type). An SDM message contains one or more **fields**. Each field consists of:

- A name, limited to 255 characters in length. Field names are *not* case-sensitive. So, "price" is the same as "Price" is the same as "PRICE".
- A type (discussed below).
- A value (particular to the field type). Each named field may only appear once in a message. If multiple fields of the same name and type are needed, create an array field. A field in a nested message **may** have the same name as a field in the outer message, though.

## Field types

The following field types (and arrays thereof) are supported by SDM:

Description	SDM Type	C Type
Boolean	<code>LBMSDM_TYPE_- BOOLEAN</code>	<code>uint8_t</code>
8-bit signed integer	<code>LBMSDM_TYPE_- INT8</code>	<code>int8_t</code>
8-bit unsigned integer	<code>LBMSDM_TYPE_- UINT8</code>	<code>uint8_t</code>
16-bit signed integer	<code>LBMSDM_TYPE_- INT16</code>	<code>int16_t</code>
16-bit unsigned integer	<code>LBMSDM_TYPE_- UINT16</code>	<code>uint16_t</code>
32-bit signed integer	<code>LBMSDM_TYPE_- INT32</code>	<code>int32_t</code>
32-bit unsigned integer	<code>LBMSDM_TYPE_- UINT32</code>	<code>uint32_t</code>
64-bit signed integer	<code>LBMSDM_TYPE_- INT64</code>	<code>int64_t</code>
64-bit unsigned integer	<code>LBMSDM_TYPE_- UINT64</code>	<code>uint64_t</code>
Single-precision floating point	<code>LBMSDM_TYPE_- FLOAT</code>	<code>float</code>
Double-precision floating point	<code>LBMSDM_TYPE_- DOUBLE</code>	<code>double</code>
String	<code>LBMSDM_TYPE_- STRING</code>	<code>char *</code>
Scaled decimal	<code>LBMSDM_TYPE_- DECIMAL</code>	<code>lbmsdm_decimal_t</code>
Timestamp	<code>LBMSDM_TYPE_- TIMESTAMP</code>	<code>struct timeval</code>
Nested message	<code>LBMSDM_TYPE_- MESSAGE</code>	<code>lbmsdm_msg_t *</code>
Binary large object (BLOB)	<code>LBMSDM_TYPE_- BLOB</code>	<code>void *</code>
Unicode string	<code>LBMSDM_TYPE_- UNICODE</code>	<code>wchar_t *</code>

Note that arrays are homogeneous. All elements of an array must be of the same type. An error is reported if an attempt is made to add an element of one type to an array containing elements of a different type.

### Building a message

A message must be created (via `lbmsdm_msg_create()` or `lbmsdm_msg_parse()`) before fields can be added.

Once a field exists within a message, it can be referenced in one of three ways:

- By the name associated with the field.
- By index. This refers to the sequential position of the field within a message. The first field has index 0, the second has index 1, and so forth.
- By iterator. See below for more information on iterators.

### **Adding fields to a message**

Scalar (non-array) fields are added to a message via the `lbmsdm_msg_add_<xxx>()` API functions, where `<xxx>` is the type of the field being added. See the module [Add a field to a message](#) for information on these functions.

When adding a field, data of the appropriate type must be supplied. As an example, to add a 32-bit signed integer field named "quantity" to a message:

```
int32_t quant = 50;
int rc;
rc = lbmsdm_msg_add_int32(msg, "quantity", quant);
```

Alternatively, literals may be used to specify the value. The above example could also be coded as:

```
rc = lbmsdm_msg_add_int32(msg, "quantity", 50);
```

### **Adding array fields to a message**

Array fields are added to a message in two steps. First, the field itself is added via the `lbmsdm_msg_add_xxx_array()` API functions, where `<xxx>` is the type of the field being added. This does not provide a value for the field. See the module [Add an array field to a message](#) for information on these functions.

Second, individual elements are added to the array field. This is done via the `lbmsdm_msg_add_xxx_elem_idx()`, `lbmsdm_msg_add_xxx_-elem_name()`, and `lbmsdm_iter_add_xxx_elem()` API functions. See the modules [Add an element to an array field by field index](#), [Add an element to an array field by field name](#), and [Add an element to an array field referenced by an iterator](#) for detailed information on these functions.

As an example, the following code illustrates how to create a string array field, and add 3 elements to it.

```
rc = lbmsdm_msg_add_string_array(msg, "string_array");
rc = lbmsdm_msg_add_string_elem_name(msg, "string_array", "String1");
rc = lbmsdm_msg_add_string_elem_name(msg, "string_array", "String2");
rc = lbmsdm_msg_add_string_elem_name(msg, "string_array", "String3");
```

### Serializing the message

Once the SDM message is constructed, it must be serialized for transmission. The API function [lbmsdm\\_msg\\_get\\_data\(\)](#) returns a static pointer to a buffer containing the serialized form of the message, suitable for transmission. The length of the serialized data may be obtained via the API function [lbmsdm\\_msg\\_get\\_datalen\(\)](#). For example, a constructed message may be sent by:

```
rc = lbm_src_send(src, lbmsdm_msg_get_data(msg), lbmsdm_msg_get_length(msg), 0);
```

The pointer returned by [lbmsdm\\_msg\\_get\\_data\(\)](#) is owned by the SDM API, and will automatically be freed when the message is destroyed.

### Deserializing a message

When a message is received, it must be deserialized so that individual fields can be accessed. This is done via the [lbmsdm\\_msg\\_parse\(\)](#) API function:

```
lbmsdm_msg_t * sdmmsg;  
rc = lbmsdm_msg_parse(&sdmmsg, lbmmsg->data, lbmmsg->len);
```

### Disposing of a message

Once an SDM message (created by either the [lbmsdm\\_msg\\_create\(\)](#) or [lbmsdm\\_msg\\_parse\(\)](#) API calls) is no longer needed, it must be disposed of to avoid a resource leak. This is done via the [lbmsdm\\_msg\\_destroy\(\)](#) API call.

### Retrieving field information

A number of API functions are available to retrieve information about individual fields.

- [lbmsdm\\_msg\\_get\\_fldcnt\(\)](#) returns the number of fields in the message.
- [lbmsdm\\_msg\\_get\\_name\\_idx\(\)](#) and [lbmsdm\\_iter\\_get\\_name\(\)](#) return the field name associated with the referenced field.
- [lbmsdm\\_msg\\_get\\_type\\_name\(\)](#), [lbmsdm\\_msg\\_get\\_type\\_idx\(\)](#), and [lbmsdm\\_iter\\_get\\_type\(\)](#) return the type of the referenced field.
- [lbmsdm\\_msg\\_get\\_elemcnt\\_name\(\)](#), [lbmsdm\\_msg\\_get\\_elemcnt\\_idx\(\)](#), and [lbmsdm\\_iter\\_get\\_elemcnt\(\)](#) return the number of elements in an array field.
- [lbmsdm\\_msg\\_get\\_len\\_name\(\)](#), [lbmsdm\\_msg\\_get\\_len\\_idx\(\)](#), and [lbmsdm\\_iter\\_get\\_len\(\)](#) return the length (in bytes) required for a field.
- [lbmsdm\\_msg\\_get\\_elemlen\\_name\(\)](#), [lbmsdm\\_msg\\_get\\_elemlen\\_idx\(\)](#), and [lbmsdm\\_iter\\_get\\_elemlen\(\)](#) return the length (in bytes) for a specific element in an array field.

### Fetching fields from a message

When fetching a field from a message, the field may be referenced by name, by index, or via an iterator.

Scalar (non-array) fields may be retrieved via the `lbmsdm_msg_get_xxx_name()`, `lbmsdm_msg_get_xxx_idx()`, or `lbmsdm_iter_get_xxx()` functions, where `xxx` is the type the field value should be retrieved as. The sections [Get scalar field values by field name](#), [Get scalar field values by field index](#), and [Get a scalar field via an iterator](#) contain detailed information on these functions.

Array field elements may be retrieved via the `lbmsdm_msg_get_xxx_elem_name()`, `lbmsdm_msg_get_xxx_elem_idx()`, or `lbmsdm_iter_get_xxx_elem()` functions, where `xxx` is the type the field element value should be retrieved as. The sections [Get an element from an array field by field name](#), [Get an element from an array field by field index](#), and [Get an element from an array field referenced by an iterator](#) contain detailed information on these functions.

## Type conversion

A limited form of automatic type conversion is provided. For example, given a field defined as `LBMSDM_TYPE_UINT16`, its value may be retrieved as an `LBMSDM_TYPE_UINT32`. The following table details which type conversions are supported.

To	bool	int8	uint8	nt16	int16	uint32	int32	float	double	string	ni-time	BL	OB	mes-dec-	sagei-	mal
From	bool	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes
int8	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes
uint8	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes
int16	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes
uint16	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes
int32	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes
uint32	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes
int64	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes
uint64	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes
float	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	No
double	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	No
string	No	No	No	No	No	No	No	No	No	No	No	Yes	No	No	No	No
unicode	No	No	No	No	No	No	No	No	No	No	No	No	Yes	No	No	No
time	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes	No	No
tamp																
BL	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes	No	No
OB	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No
mes	No	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes	No
sage																
i-																
mal																

The above conversion rules apply also when retrieving array elements.

### Fetching string, unicode, and BLOB values

When fetching a field or array element value as a string, unicode, or BLOB, the data is copied into a buffer provided by the application. In addition to the buffer, the size of the buffer must be given. The size is specified in bytes for string and BLOB fields, and in wchar\_ts for unicode fields. If the size specified is too small for the data, the error code [LBMSDM\\_INSUFFICIENT\\_BUFFER\\_LENGTH](#) is returned.

### Fetching message fields

When fetching the value of a message field, a copy of the message is created (via [lbmsdm\\_msg\\_clone\(\)](#)) and returned. It is the application's responsibility to destroy the message (via [lbmsdm\\_msg\\_destroy\(\)](#)) when it is no longer needed.

### Modifying fields in a message

Existing fields in a message may be modified, both in terms of the field type and field value. For scalar (non-array) fields, the `lbmsdm_msg_set_xxx_idx()`, `lbmsdm_msg_set_xxx_name()`, and `lbmsdm_iter_set_xxx()` API functions may be used, where `xxx` is the type to be assigned to the field. See the sections [Set a field value in a message by field index](#), [Set a field value in a message by field name](#), and [Set a field value in a message referenced by an iterator](#) for information on these functions.

For array fields, the `lbmsdm_msg_set_xxx_array_idx()`, `lbmsdm_msg_set_xxx_array_name()`, and `lbmsdm_iter_set_xxx_array()` API functions may be used, where `xxx` is the type to be assigned to the field. See the sections [Set a field value in a message by field index to an array field](#), [Set a field value in a message by field name to an array field](#), and [Set a field value in a message, referenced by an iterator, to an array field](#). for information on these functions. As when adding an array field to a message, once the field type has been set to an array type, individual elements must be added to the array field.

Individual elements of an array field may be modified via the `lbmsdm_msg_set_xxx_elem_idx()`, `lbmsdm_msg_set_xxx_elem_name()`, and `lbmsdm_iter_set_xxx_elem()` API functions. See the sections [Set an array field element value by field index](#), [Set an array field element value by field name](#), and [Set an array field element value for a field referenced by an iterator](#) for

information on these functions. Note that arrays must contain homogeneous elements, so the type of an array element may not be changed, and is considered an error.

### Deleting fields from a message

A field may be deleted from a message via the [lbmsdm\\_msg\\_del\\_idx\(\)](#), [lbmsdm\\_msg\\_del\\_name\(\)](#), and [lbmsdm\\_iter\\_del\(\)](#) API calls. Deleting a field will cause any fields following it to be moved down one position, changing the index of those fields and potentially invalidating any iterators for that message.

### Deleting elements from an array field

Individual elements may be deleted from an array field via the [lbmsdm\\_msg\\_del\\_elem\\_idx\(\)](#), [lbmsdm\\_msg\\_del\\_elem\\_name\(\)](#), and [lbmsdm\\_iter\\_del\\_elem\(\)](#) API functions.

### Null fields

SDM supports the concept of a **null** field. A null field is present in the message, but has no value associated with it. Once added to a message, a field may be set to null via the [lbmsdm\\_msg\\_set\\_null\\_idx\(\)](#), [lbmsdm\\_msg\\_set\\_null\\_name\(\)](#), or [lbmsdm\\_iter\\_set\\_null\(\)](#) API functions. Setting the field (which may be either a scalar or array field) to null removes any values currently associated with the field.

The [lbmsdm\\_msg\\_is\\_null\\_idx\(\)](#), [lbmsdm\\_msg\\_is\\_null\\_name\(\)](#), and [lbmsdm\\_iter\\_is\\_null\(\)](#) API functions allow an application to determine if a given field is null.

Attempting to retrieve a value or element value from a null field is not allowed, and will return an error.

### Iterators

A field iterator allows sequential operation on the fields of a message without requiring the field name or index. An iterator is created via [lbmsdm\\_iter\\_create\(\)](#), the first field in a message is located via [lbmsdm\\_iter\\_first\(\)](#), and the next field is located via [lbmsdm\\_iter\\_next\(\)](#). An iterator should be destroyed when no longer needed, using the [lbmsdm\\_iter\\_destroy\(\)](#) API call.

Message fields may be queried, fetched, modified, and deleted via an iterator. In each case, the operation applies to the field currently referenced by the iterator.

### Error information

All functions return a value to indicate the success or failure of the operation. Most return [LBMSDM\\_SUCCESS](#) to indicate success, or [LBMSDM\\_FAILURE](#) otherwise. Consult the individual function documentation for exceptions.

The function [lbmsdm\\_errnum\(\)](#) can be used to retrieve a detailed error code for the last error encountered, while [lbmsdm\\_errmsg\(\)](#) will return a descriptive error message.

## Message Options

The performance of SDM can be tuned through the use of message options. Options are contained within an attributes object ([lbmsdm\\_msg\\_attr\\_t](#)), which is created via [lbmsdm\\_msg\\_attr\\_create\(\)](#). When no longer needed, an attributes object can be discarded by calling [lbmsdm\\_msg\\_attr\\_delete\(\)](#). Individual options within an attributes object can be set via [lbmsdm\\_msg\\_attr\\_setopt\(\)](#) and [lbmsdm\\_msg\\_attr\\_str\\_setopt\(\)](#), and can be queried via [lbmsdm\\_msg\\_attr\\_getopt\(\)](#) and [lbmsdm\\_msg\\_attr\\_str\\_getopt\(\)](#). A set of options can be specified at message creation time using [lbmsdm\\_msg\\_create\\_ex\(\)](#) and [lbmsdm\\_msg\\_parse\\_ex\(\)](#).

The following table lists the supported message options.

Option	Data type	Allowed values	Default	Description
field_array_allocation	int	Any integer $\geq 0$	32	<p>Internally, SDM maintains an array of field entries within a message. This option controls both the number of field entries initially allocated when the message is created, and the increment used when the array must be expanded when a field is added but the array is full.</p> <p>If it is known that a large number of fields will be added to a message, setting this option to a larger value will result in a slight performance boost, since reallocation of the field array will occur less frequently.</p>
Generated on Thu Mar 6 13:11:08 2014 for LBM API by Doxygen				Similarly, if the number of fields to be added is small, setting this option to a smaller value will

As an example, the following code fragment creates an attributes object, sets options, get the options, creates a message using the attributes object, then destroys the attributes object. For the sake of brevity, error checking has been omitted, as has the code to add fields to the message.

```
lbmsdm_msg_attr_t * attr;
lbmsdm_msg_t * msg;
int name_tree;
int alloc_size;
char val_buf[256];
size_t val_len;

lbmsdm_msg_attr_create(&attr);

name_tree = 0;
lbmsdm_msg_attr_setopt(attr, "name_tree", (void *)&name_tree, sizeof(name_tree));
lbmsdm_msg_attr_str_setopt(attr, "field_array_allocation", "128");

val_len = sizeof(val_buf);
lbmsdm_msg_attr_str_getopt(attr, "name_tree", val_buf, &val_len);
printf("name_tree=%s\n", val_buf);
val_len = sizeof(alloc_size);
lbmsdm_msg_attr_getopt(attr, "field_array_allocation", (void *)&alloc_size, &val_len);
printf("field_array_allocation=%d\n", alloc_size);

lbmsdm_msg_create_ex(&msg, attr);

lbmsdm_msg_attr_delete(attr);
```

## 8.6.2 Typedef Documentation

### 8.6.2.1 `typedef struct lbmsdm_decimal_t_stct lbmsdm_decimal_t`

The mantissa is represented as a 64-bit signed integer. The exponent is represented as an 8-bit signed integer, and can range from -128 to 127.

## 8.6.3 Enumeration Type Documentation

### 8.6.3.1 `anonymous enum`

**Enumerator:**

***LBMSDM\_TYPE\_INVALID*** SDM field type: Type is invalid.

***LBMSDM\_TYPE\_BOOLEAN*** SDM field type: Boolean (non-zero is true, zero is false).

***LBMSDM\_TYPE\_INT8*** SDM field type: 8-bit signed integer.

***LBMSDM\_TYPE\_UINT8*** SDM field type: 8-bit unsigned integer.

***LBMSDM\_TYPE\_INT16*** SDM field type: 16-bit signed integer.

***LBMSDM\_TYPE\_UINT16*** SDM field type: 16-bit unsigned integer.

***LBMSDM\_TYPE\_INT32*** SDM field type: 32-bit signed integer.

***LBMSDM\_TYPE\_UINT32*** SDM field type: 32-bit unsigned integer.

***LBMSDM\_TYPE\_INT64*** SDM field type: 64-bit signed integer.

***LBMSDM\_TYPE\_UINT64*** SDM field type: 64-bit unsigned integer.

***LBMSDM\_TYPE\_FLOAT*** SDM field type: Single-precision floating point.

***LBMSDM\_TYPE\_DOUBLE*** SDM field type: Double-precision floating point.

***LBMSDM\_TYPE\_DECIMAL*** SDM field type: Decimal number.

***LBMSDM\_TYPE\_TIMESTAMP*** SDM field type: Seconds and microseconds since the epoch (UTC).

***LBMSDM\_TYPE\_MESSAGE*** SDM field type: Nested SDM message.

***LBMSDM\_TYPE\_STRING*** SDM field type: Character string (ASCIZ).

***LBMSDM\_TYPE\_UNICODE*** SDM field type: Unicode string.

***LBMSDM\_TYPE\_BLOB*** SDM field type: Binary Large Object (BLOB).

***LBMSDM\_TYPE\_ARRAY\_BOOLEAN*** SDM field type: Array of Booleans (non-zero is true, zero is false).

***LBMSDM\_TYPE\_ARRAY\_INT8*** SDM field type: Array of 8-bit signed integers.

***LBMSDM\_TYPE\_ARRAY\_UINT8*** SDM field type: Array of 8-bit unsigned integers.

***LBMSDM\_TYPE\_ARRAY\_INT16*** SDM field type: Array of 16-bit signed integers.

***LBMSDM\_TYPE\_ARRAY\_UINT16*** SDM field type: Array of 16-bit unsigned integers.

***LBMSDM\_TYPE\_ARRAY\_INT32*** SDM field type: Array of 32-bit signed integers.

***LBMSDM\_TYPE\_ARRAY\_UINT32*** SDM field type: Array of 32-bit unsigned integers.

***LBMSDM\_TYPE\_ARRAY\_INT64*** SDM field type: Array of 64-bit signed integers.

***LBMSDM\_TYPE\_ARRAY\_UINT64*** SDM field type: Array of 64-bit unsigned integers.

***LBMSDM\_TYPE\_ARRAY\_FLOAT*** SDM field type: Array of single-precision floating points.

***LBMSDM\_TYPE\_ARRAY\_DOUBLE*** SDM field type: Array of double-precision floating points.

***LBMSDM\_TYPE\_ARRAY\_DECIMAL*** SDM field type: Array of decimal numbers.

***LBMSDM\_TYPE\_ARRAY\_TIMESTAMP*** SDM field type: Array of timestamps (seconds and microseconds since the epoch (UTC)).

***LBMSDM\_TYPE\_ARRAY\_MESSAGE*** SDM field type: Array of nested SDM messages.

***LBMSDM\_TYPE\_ARRAY\_STRING*** SDM field type: Array of character strings (ASCIZ).

***LBMSDM\_TYPE\_ARRAY\_UNICODE*** SDM field type: Array of unicode strings.

***LBMSDM\_TYPE\_ARRAY\_BLOB*** SDM field type: Array of Binary Large Objects (BLOB).

#### 8.6.3.2 anonymous enum

Enumerator:

***LBMSDM\_SUCCESS*** SDM return code: Operation was successful.

***LBMSDM\_FAILURE*** SDM return code: Operation failed. See [lbmsdm\\_errnum\(\)](#) or [lbmsdm\\_errmsg\(\)](#) for the reason.

***LBMSDM\_FIELD\_IS\_NULL*** SDM return code: Field is null.

***LBMSDM\_NO\_MORE\_FIELDS*** SDM return code: No more fields to iterate over.

***LBMSDM\_INSUFFICIENT\_BUFFER\_LENGTH*** SDM return code: Insufficient buffer length given.

#### 8.6.3.3 anonymous enum

Enumerator:

***LBMSDM\_ERR\_EINVAL*** SDM error code: An invalid argument was passed.

***LBMSDM\_ERR\_ENOMEM*** SDM error code: Operation could not be completed due to memory allocation error.

***LBMSDM\_ERR\_NAMETOOLONG*** SDM error code: Field name is too long.

***LBMSDM\_ERR\_DUPLICATE\_FIELD*** SDM error code: The field being added to the message already exists.

***LBMSDM\_ERR\_BAD\_TYPE*** SDM error code: Invalid type.

***LBMSDM\_ERR\_FIELD\_NOT\_FOUND*** SDM error code: The field does not exist in the message.

***LBMSDM\_ERR\_MSG\_INVALID*** SDM error code: The message is in an invalid form.

***LBMSDM\_ERR\_CANNOT\_CONVERT*** SDM error code: The field can not be converted as requested.

***LBMSDM\_ERR\_NOT\_ARRAY*** SDM error code: The field is not an array field.

***LBMSDM\_ERR\_NOT\_SCALAR*** SDM error code: The field is not a scalar field.

***LBMSDM\_ERR\_ELEMENT\_NOT\_FOUND*** SDM error code: The specified array element does not exist.

***LBMSDM\_ERR\_TYPE\_NOT\_SUPPORTED*** SDM error code: The specified type is not supported.

***LBMSDM\_ERR\_TYPE\_MISMATCH*** SDM error code: Type mismatch.

***LBMSDM\_ERR\_UNICODE\_CONVERSION*** SDM error code: Unicode conversion error.

***LBMSDM\_ERR\_FIELD\_IS\_NULL*** SDM error code: Field is null.

***LBMSDM\_ERR\_ADDING\_FIELD*** SDM error code: Unable to add field.

***LBMSDM\_ERR\_ITERATOR\_INVALID*** SDM error code: Iterator doesn't reference a valid field.

***LBMSDM\_ERR\_DELETING\_FIELD*** SDM error code: Error deleting a field.

***LBMSDM\_ERR\_INVALID\_FIELD\_NAME*** SDM error code: Invalid field name.

#### 8.6.4 Function Documentation

##### 8.6.4.1 LBMSDMExpDLL const char\* lbmsdm\_errmsg (void)

###### Returns:

Pointer to a static char array containing the error message.

##### 8.6.4.2 LBMSDMExpDLL int lbmsdm\_errnum (void)

###### Returns:

Integer error number (see LBMSDM\_ERROR\_\*).

##### 8.6.4.3 LBMSDMExpDLL int lbmsdm\_iter\_create (lbmsdm\_iter\_t \*\* *Iterator*, lbmsdm\_msg\_t \* *Message*)

###### Parameters:

*Iterator* A pointer to a pointer to an SDM iterator object. Will be filled in by this function to point to the newly created `lbmsdm_iter_t` object.

**Message** SDM message on which the iterator is to operate.

**Returns:**

`LBMSDM_SUCCESS` if successful, `LBMSDM_FAILURE` otherwise.

#### 8.6.4.4 LBMSDMExpDLL int lbmsdm\_iter\_del (lbmsdm\_iter\_t \* *Iterator*)

**Parameters:**

*Iterator* The SDM iterator to use.

**Returns:**

`LBMSDM_SUCCESS` if successful, `LBMSDM_FAILURE` otherwise.

#### 8.6.4.5 LBMSDMExpDLL int lbmsdm\_iter\_del\_elem (lbmsdm\_iter\_t \* *Iterator*, size\_t *Element*)

**Parameters:**

*Iterator* The SDM iterator to use.

*Element* Element to be deleted.

**Returns:**

`LBMSDM_SUCCESS` if successful, `LBMSDM_FAILURE` otherwise.

#### 8.6.4.6 LBMSDMExpDLL int lbmsdm\_iter\_destroy (lbmsdm\_iter\_t \* *Iterator*)

**Parameters:**

*Iterator* The `lbmsdm_iter_t` object to destroy.

**Returns:**

`LBMSDM_SUCCESS` if successful, `LBMSDM_FAILURE` otherwise.

#### 8.6.4.7 LBMSDMExpDLL int lbmsdm\_iter\_first (lbmsdm\_iter\_t \* *Iterator*)

**Parameters:**

*Iterator* The iterator to position.

**Return values:**

- LBMSDM\_SUCCESS* if successful
- LBMSDM\_NO\_MORE\_FIELDS* if no fields exist in the message
- LBMSDM\_FAILURE* if the operation failed

**8.6.4.8 LBMSDMExpDLL int lbmsdm\_iter\_get\_element (lbmsdm\_iter\_t \*  
*Iterator*)****Parameters:**

*Iterator* The SDM iterator.

**Returns:**

The number of elements in the array, or -1 if an error occurred.

**Note:**

Calling this function for a non-array field will return 1 for the number of elements.

**8.6.4.9 LBMSDMExpDLL int lbmsdm\_iter\_get\_elemlen (lbmsdm\_iter\_t \*  
*Iterator*, size\_t *Element*)****Parameters:**

*Iterator* The SDM iterator.

*Element* Element index (zero-based).

**Returns:**

The number of bytes required to store the field, or -1 if an error occurred.

**8.6.4.10 LBMSDMExpDLL int lbmsdm\_iter\_get\_len (lbmsdm\_iter\_t \**Iterator*)****Parameters:**

*Iterator* The SDM iterator.

**Returns:**

The number of bytes required to store the field, or -1 if an error occurred.

**Note:**

Calling this function for an array field will return -1.

**See also:**

`lbmsdm_iter_get_field_array_size_index_elem()`

**8.6.4.11 LBMSDMExpDLL const char\* lbmsdm\_iter\_get\_name  
(lbmsdm\_iter\_t \* *Iterator*)****Parameters:**

*Iterator* The SDM iterator.

**Returns:**

The field name, or NULL if an error occurred.

**8.6.4.12 LBMSDMExpDLL lbmsdm\_field\_type\_t lbmsdm\_iter\_get\_type  
(lbmsdm\_iter\_t \* *Iterator*)****Parameters:**

*Iterator* The SDM iterator.

**Returns:**

The field type, or [LBMSDM\\_TYPE\\_INVALID](#) if an error occurred.

**8.6.4.13 LBMSDMExpDLL int lbmsdm\_iter\_is\_null (lbmsdm\_iter\_t \* *Iterator*)****Parameters:**

*Iterator* The SDM iterator.

**Return values:**

*1* if the field is null.

*0* if the field is present and not null.

[LBMSDM\\_FAILURE](#) if an error occurred.

**8.6.4.14 LBMSDMExpDLL int lbmsdm\_iter\_next (lbmsdm\_iter\_t \* *Iterator*)****Parameters:**

*Iterator* The iterator to position.

**Return values:**

- LBMSDM\_SUCCESS* if successful
- LBMSDM\_NO\_MORE\_FIELDS* if no fields exist in the message
- LBMSDM\_FAILURE* if the operation failed

**8.6.4.15 LBMSDMDExpDLL int lbmsdm\_iter\_set\_null (lbmsdm\_iter\_t \*  
*Iterator*)****Parameters:**

*Iterator* The SDM iterator.

**Returns:**

*LBMSDM\_SUCCESS* if successful, *LBMSDM\_FAILURE* otherwise.

**8.6.4.16 LBMSDMDExpDLL int lbmsdm\_msg\_attr\_create (lbmsdm\_msg\_attr\_t  
\*\* *Attributes*)**

The attribute object is allocated and filled in with the default values that are used by *lbmsdm\_msg\_t* objects.

**Parameters:**

*Attributes* Pointer to a pointer to an SDM message attribute structure. Will be filled in by this function to point to the newly created *lbmsdm\_msg\_attr\_t* object.

**Returns:**

*LBMSDM\_SUCCESS* if successful, *LBMSDM\_FAILURE* otherwise.

**8.6.4.17 LBMSDMDExpDLL int lbmsdm\_msg\_attr\_delete (lbmsdm\_msg\_attr\_t  
\* *Attributes*)**

The attribute object is cleaned up and deleted.

**Parameters:**

*Attributes* Pointer to an SDM message attribute structure as returned by *lbmsdm\_msg\_attr\_create* or *lbmsdm\_msg\_attr\_dup*.

**Returns:**

*LBMSDM\_SUCCESS* if successful, *LBMSDM\_FAILURE* otherwise.

**8.6.4.18 LBMSDMExpDLL int lbmsdm\_msg\_attr\_dup (lbmsdm\_msg\_attr\_t \*Attributes, lbmsdm\_msg\_attr\_t \*Original)**

A new attribute object is created as a copy of an existing object.

**Parameters:**

*Attributes* Pointer to a pointer to an SDM message attribute structure. Will be filled in by this function to point to the newly created `lbmsdm_msg_attr_t` object.

*Original* Pointer to an SDM message attribute structure as returned by `lbmsdm_msg_attr_create` or `lbmsdm_msg_attr_dup`, from which *Attributes* is initialized.

**Returns:**

`LBMSDM_SUCCESS` if successful, `LBMSDM_FAILURE` otherwise.

**8.6.4.19 LBMSDMExpDLL int lbmsdm\_msg\_attr\_getopt (lbmsdm\_msg\_attr\_t \*Attributes, const char \*Option, void \*Value, size\_t \*Length)****Parameters:**

*Attributes* Pointer to an SDM message attribute structure.

*Option* String containing the option name.

*Value* Pointer to the option value structure to be filled. The structure of the option value is specific to the option itself.

*Length* Length (in bytes) of the *Value* structure when passed in. Upon return, this is set to the actual size of the filled-in structure.

**Returns:**

`LBMSDM_SUCCESS` if successful, `LBMSDM_FAILURE` otherwise.

**8.6.4.20 LBMSDMExpDLL int lbmsdm\_msg\_attr\_setopt (lbmsdm\_msg\_attr\_t \*Attributes, const char \*Option, void \*Value, size\_t Length)**

Used before the message is created. NOTE: the attribute object must first be created with `lbmsdm_msg_attr_create` or `lbmsdm_msg_attr_dup`.

**Parameters:**

*Attributes* Pointer to an SDM message attribute structure.

*Option* String containing the option name.

**Value** Pointer to the option value structure. The structure of the option value is specific to the option itself.

**Length** Length (in bytes) of the structure.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

**8.6.4.21 LBMSDMDExpDLL int lbmsdm\_msg\_attr\_str\_getopt  
(lbmsdm\_msg\_attr\_t \*Attributes, const char \*Option, char \*Value,  
size\_t \*Length)**

**Parameters:**

**Attributes** Pointer to an SDM message attribute structure.

**Option** String containing the option name.

**Value** Pointer to the string to be filled in.

**Length** Maximum length (in bytes) of the *Value* string when passed in. Upon return, this is set to the size of the formatted string.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

**8.6.4.22 LBMSDMDExpDLL int lbmsdm\_msg\_attr\_str\_setopt  
(lbmsdm\_msg\_attr\_t \*Attributes, const char \*Option, const char \*  
Value)**

Used before the message is created. NOTE: the attribute object must first be created with [lbmsdm\\_msg\\_attr\\_create](#) or [lbmsdm\\_msg\\_attr\\_dup](#).

**Parameters:**

**Attributes** Pointer to an SDM message attribute structure.

**Option** String containing the option name.

**Value** String containing the option value. The format of the string is specific to the option itself.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

**8.6.4.23 LBMSDMDLL int lbmsdm\_msg\_clear (lbmsdm\_msg\_t \* *Message*)****Parameters:**

*Message* The SDM message to clear.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

**8.6.4.24 LBMSDMDLL int lbmsdm\_msg\_clone (lbmsdm\_msg\_t \*\*  
*Message*, const lbmsdm\_msg\_t \* *Original*)****Parameters:**

*Message* A pointer to a pointer to an SDM message object. Will be filled in by this function to point to the newly created [lbmsdm\\_msg\\_t](#) object.

*Original* The SDM message to be cloned.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

**8.6.4.25 LBMSDMDLL int lbmsdm\_msg\_create (lbmsdm\_msg\_t \*\*  
*Message*)****Parameters:**

*Message* A pointer to a pointer to an SDM message object. Will be filled in by this function to point to the newly created [lbmsdm\\_msg\\_t](#) object.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

**8.6.4.26 LBMSDMDLL int lbmsdm\_msg\_create\_ex (lbmsdm\_msg\_t \*\*  
*Message*, const lbmsdm\_msg\_attr\_t \* *Attributes*)****Parameters:**

*Message* A pointer to a pointer to an SDM message object. Will be filled in by this function to point to the newly created [lbmsdm\\_msg\\_t](#) object.

*Attributes* A pointer to an [lbmsdm\\_msg\\_attr\\_t](#) structure used to initialize the message options.

**Returns:**

`LBMSDM_SUCCESS` if successful, `LBMSDM_FAILURE` otherwise.

**8.6.4.27 LBMSDMDExpDLL int lbmsdm\_msg\_del\_elem\_idx (lbmsdm\_msg\_t \*  
\* *Message*, size\_t *Index*, size\_t *Element*)****Parameters:**

*Message* The SDM message containing the field.

*Index* Field index.

*Element* Element to be deleted.

**Returns:**

`LBMSDM_SUCCESS` if successful, `LBMSDM_FAILURE` otherwise.

**8.6.4.28 LBMSDMDExpDLL int lbmsdm\_msg\_del\_elem\_name (lbmsdm\_msg\_t  
\* *Message*, const char \* *Name*, size\_t *Element*)****Parameters:**

*Message* The SDM message containing the field.

*Name* Field name.

*Element* Element to be deleted.

**Returns:**

`LBMSDM_SUCCESS` if successful, `LBMSDM_FAILURE` otherwise.

**8.6.4.29 LBMSDMDExpDLL int lbmsdm\_msg\_del\_idx (lbmsdm\_msg\_t \*  
\* *Message*, size\_t *Index*)****Parameters:**

*Message* The SDM message containing the field.

*Index* Field index.

**Returns:**

`LBMSDM_SUCCESS` if successful, `LBMSDM_FAILURE` otherwise.

**8.6.4.30 LBMSDMEExpDLL int lbmsdm\_msg\_del\_name (lbmsdm\_msg\_t \*  
Message, const char \* Name)**

**Parameters:**

*Message* The SDM message containing the field.

*Name* Field name.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

**8.6.4.31 LBMSDMEExpDLL int lbmsdm\_msg\_destroy (lbmsdm\_msg\_t \*  
Message)**

**Parameters:**

*Message* The SDM message to destroy.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

**8.6.4.32 LBMSDMEExpDLL int lbmsdm\_msg\_dump (lbmsdm\_msg\_t \*  
Message, char \* Buffer, size\_t Size)**

**Parameters:**

*Message* The SDM message to dump.

*Buffer* Buffer into which to dump the message.

*Size* Maximum size of *Buffer*.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

**Note:**

*Buffer* will be null-terminated. If *Buffer* isn't large enough to contain the entire message, it will be truncated to fit the available space. The values for unicode and BLOB fields will not be formatted.

**8.6.4.33 LBMSDMExpDLL const char\* lbmsdm\_msg\_get\_data  
(lbmsdm\_msg\_t \* *Message*)****Parameters:**

*Message* The SDM message.

**Returns:**

A pointer to the data buffer, or NULL if any error occurs.

**Note:**

The pointer returned by `lbmsdm_msg_get_data` is invalidated when the message is deleted (via `lbmsdm_msg_destroy`), any field is added to the message, any field is deleted from the message, or any field in the message is changed (either the field value or type). In other words, any time the message is changed, the pointer returned is no longer valid.

**8.6.4.34 LBMSDMExpDLL size\_t lbmsdm\_msg\_get\_datalen (lbmsdm\_msg\_t \*  
*Message*)****Parameters:**

*Message* The SDM message.

**Returns:**

The length of the data buffer, or 0 if any error occurs.

**8.6.4.35 LBMSDMExpDLL int lbmsdm\_msg\_get\_element\_idx (lbmsdm\_msg\_t  
\* *Message*, size\_t *Index*)****Parameters:**

*Message* The SDM message from which the array size is to be fetched.

*Index* Field index.

**Returns:**

The number of elements in the array, or -1 if an error occurred.

**Note:**

Calling this function for a non-array field will return 1 for the number of elements.

**8.6.4.36 LBMSDMEExpDLL int lbmsdm\_msg\_get\_element\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*)****Parameters:**

*Message* The SDM message from which the array size is to be fetched.

*Name* Field name.

**Returns:**

The number of elements in the array, or -1 if an error occurred.

**Note:**

Calling this function for a non-array field will return 1 for the number of elements.

**8.6.4.37 LBMSDMEExpDLL int lbmsdm\_msg\_get\_elemlen\_idx (lbmsdm\_msg\_t  
\* *Message*, size\_t *Index*, size\_t *Element*)****Parameters:**

*Message* The SDM message containing the field.

*Index* Field index.

*Element* Element index (zero-based).

**Returns:**

The number of bytes required to store the field, or -1 if an error occurred.

**8.6.4.38 LBMSDMEExpDLL int lbmsdm\_msg\_get\_elemlen\_name  
(lbmsdm\_msg\_t \* *Message*, const char \* *Name*, size\_t *Element*)****Parameters:**

*Message* The SDM message containing the field.

*Name* Field name.

*Element* Element index (zero-based).

**Returns:**

The number of bytes required to store the field, or -1 if an error occurred.

**8.6.4.39 LBMSDMExpDLL int lbmsdm\_msg\_get\_fldent (lbmsdm\_msg\_t \*  
*Message*)**

**Parameters:**

*Message* The SDM message.

**Returns:**

The number of fields in the message, or -1 if an error occurs.

**Note:**

Only top-level fields are counted. If a message field exists within the message, the number of fields in the contained message are not counted. Instead, the message field counts as one field. Likewise, array fields contribute only 1 to the field count, not the number of elements in the field.

**8.6.4.40 LBMSDMExpDLL int lbmsdm\_msg\_get\_idx\_name (lbmsdm\_msg\_t \*  
*Message*, const char \* *Name*)**

**Parameters:**

*Message* The SDM message from which the field name is to be fetched.

*Name* Field name.

**Returns:**

The field index, or -1 if an error occurred.

**8.6.4.41 LBMSDMExpDLL int lbmsdm\_msg\_get\_len\_idx (lbmsdm\_msg\_t \*  
*Message*, size\_t *Index*)**

**Parameters:**

*Message* The SDM message containing the field.

*Index* Field index.

**Returns:**

The number of bytes required to store the field, or -1 if an error occurred.

**Note:**

Calling this function for an array field will return -1.

**See also:**

`lbmsdm_msg_get_field_array_size_index_elem()`

**8.6.4.42 LBMSDMDExpDLL int lbmsdm\_msg\_get\_len\_name (lbmsdm\_msg\_t \*  
*Message*, const char \**Name*)****Parameters:**

*Message* The SDM message containing the field.

*Name* Field name.

**Returns:**

The number of bytes required to store the field, or -1 if an error occurred.

**Note:**

Calling this function for an array field will return -1.

**See also:**

`lbmsdm_msg_get_field_array_size_name_elem()`

**8.6.4.43 LBMSDMDExpDLL const char\* lbmsdm\_msg\_get\_name\_idx  
(lbmsdm\_msg\_t \**Message*, size\_t *Index*)****Parameters:**

*Message* The SDM message from which the field name is to be fetched.

*Index* Field index.

**Returns:**

The field name, or NULL if an error occurred.

**8.6.4.44 LBMSDMDExpDLL lbmsdm\_field\_type\_t lbmsdm\_msg\_get\_type\_idx  
(lbmsdm\_msg\_t \**Message*, size\_t *Index*)****Parameters:**

*Message* The SDM message from which the field type is to be fetched.

*Index* Field index.

**Returns:**

The field type, or [LBMSDM\\_TYPE\\_INVALID](#) if an error occurred.

**8.6.4.45 LBMSDMExpDLL lbmsdm\_field\_type\_t lbmsdm\_msg\_get\_type\_name (lbmsdm\_msg\_t \* *Message*, const char \* *Name*)****Parameters:**

*Message* The SDM message from which the field type is to be fetched.

*Name* Field name.

**Returns:**

The field type, or [LBMSDM\\_TYPE\\_INVALID](#) if an error occurred.

**8.6.4.46 LBMSDMExpDLL int lbmsdm\_msg\_is\_null\_idx (lbmsdm\_msg\_t \* *Message*, size\_t *Index*)****Parameters:**

*Message* The SDM message containing the field.

*Index* Field index.

**Return values:**

*1* if the field is null.

*0* if the field is present and not null.

[LBMSDM\\_FAILURE](#) if an error occurred.

**8.6.4.47 LBMSDMExpDLL int lbmsdm\_msg\_is\_null\_name (lbmsdm\_msg\_t \* *Message*, const char \* *Name*)****Parameters:**

*Message* The SDM message containing the field.

*Name* Field name.

**Return values:**

*1* if the field is null.

*0* if the field is present and not null.

[LBMSDM\\_FAILURE](#) if an error occurred.

**8.6.4.48 LBMSDMExpDLL int lbmsdm\_msg\_parse (lbmsdm\_msg\_t \*\*  
Message, const char \* Data, size\_t Length)**

**Parameters:**

*Message* A pointer to a pointer to an SDM message object. Will be filled in by this function to point to the newly created `lbmsdm_msg_t` object.

*Data* A pointer to the buffer from which the message should be constructed.

*Length* Length of *Data*.

**Returns:**

`LBMSDM_SUCCESS` if successful, `LBMSDM_FAILURE` otherwise.

**8.6.4.49 LBMSDMExpDLL int lbmsdm\_msg\_parse\_ex (lbmsdm\_msg\_t \*\*  
Message, const char \* Data, size\_t Length, const lbmsdm\_msg\_attr\_t \*  
Attributes)**

**Parameters:**

*Message* A pointer to a pointer to an SDM message object. Will be filled in by this function to point to the newly created `lbmsdm_msg_t` object.

*Data* A pointer to the buffer from which the message should be constructed.

*Length* Length of *Data*.

*Attributes* A pointer to an `lbmsdm_msg_attr_t` structure used to initialize the message options.

**Returns:**

`LBMSDM_SUCCESS` if successful, `LBMSDM_FAILURE` otherwise.

**8.6.4.50 LBMSDMExpDLL int lbmsdm\_msg\_parse\_reuse (lbmsdm\_msg\_t \*  
Message, const char \* Data, size\_t Length)**

**Parameters:**

*Message* A pointer to an existing SDM message object, into which the message buffer is parsed. The message will be cleared before parsing.

*Data* A pointer to the buffer from which the message should be constructed.

*Length* Length of *Data*.

**Returns:**

`LBMSDM_SUCCESS` if successful, `LBMSDM_FAILURE` otherwise.

**8.6.4.51 LBMSDMExpDLL int lbmsdm\_msg\_set\_null\_idx (lbmsdm\_msg\_t \*  
*Message*, size\_t *Index*)****Parameters:**

*Message* The SDM message containing the field.

*Index* Field index.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

**8.6.4.52 LBMSDMExpDLL int lbmsdm\_msg\_set\_null\_name (lbmsdm\_msg\_t \*  
*Message*, const char \* *Name*)****Parameters:**

*Message* The SDM message containing the field.

*Name* Field name.

**Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

**8.6.4.53 LBMSDMExpDLL int lbmsdm\_win32\_static\_init (void)****Returns:**

[LBMSDM\\_SUCCESS](#) if successful, [LBMSDM\\_FAILURE](#) otherwise.

## 8.7 umeblocks.h File Reference

UME Blocking API.

### Data Structures

- struct [ume\\_sem\\_t\\_stct](#)
- struct [ume\\_block\\_src\\_t\\_stct](#)

*Structure used to designate an UME Block source.*

### Defines

- #define **SLEEP\_SEC**(x) sleep(x)
- #define **SLEEP\_MSEC**(x)
- #define **UME\_BLOCK\_DEBUG** 0
- #define **UME\_BLOCK\_DEBUG\_PRINT**(t,...)
- #define **UME\_BLOCK\_PRINT\_ERROR**(t,...)
- #define **UME\_BLOCKING\_TYPE** "Posix pthread\_cond"
- #define **UME\_SEM\_INIT**(sem, len, ret)
- #define **UME\_SEM\_POST**(sem)
- #define **UME\_SEM\_WAIT**(sem)
- #define **UME\_SEM\_GETVALUE**(sem, value) value = sem.state
- #define **UME\_SEM\_DESTROY**(sem)
- #define **UME\_SEM\_TIMEDWAIT**(sem, mstime, ret)
- #define **UME\_SEM\_TIMEDOUT**(v) 0
- #define **UME\_SEM\_TIMEDOK**(v) 0
- #define **UME\_TIMESPEC\_MSSET**(t, s, n) 0
- #define **UME\_MALLOC\_RETURN**(e, s, r)
- #define **UME\_TIME\_OUT** 5000
- #define **UME\_RETRY\_COUNT** 10

### Typedefs

- typedef [ume\\_sem\\_t\\_stct](#) **ume\_sem\_t**
- typedef [ume\\_block\\_bitmap\\_t\\_stct](#) **ume\_block\_bitmap\_t**
- typedef [ume\\_block\\_src\\_t\\_stct](#) **ume\_block\_src\_t**

*Structure used to designate an UME Block source.*

## Functions

- int `ume_block_src_delete (ume_block_src_t *asrc)`  
*Delete an UMEBlock Source object.*
- int `ume_block_src_create (ume_block_src_t **srcp, lbm_context_t *ctx, lbm_topic_t *topic, lbm_src_topic_attr_t *tattr, lbm_src_cb_proc proc, void *clientd, lbm_event_queue_t *evq)`  
*Create an UMEBlock Source that will send messages to a given topic.*
- int `ume_block_src_send_ex (ume_block_src_t *asrc, const char *msg, size_t len, int flags, lbm_src_send_ex_info_t *info)`  
*Extended send of a message to the topic associated with an UMBlock source.*

### 8.7.1 Detailed Description

The Ultra Messaging Enterprise (UME) API Description.

### 8.7.2 Define Documentation

#### 8.7.2.1 #define SLEEP\_MSEC(x)

**Value:**

```
do{ \
    if ((x) >= 1000){ \
        sleep((x) / 1000); \
        usleep((x) % 1000 * 1000); \
    } \
    else{ \
        usleep((x)*1000); \
    } \
}while (0)
```

#### 8.7.2.2 #define UME\_BLOCK\_DEBUG\_PRINT(t,...)

**Value:**

```
do { \
    if(UME_BLOCK_DEBUG) { \
        fprintf(stderr, t, ##__VA_ARGS__); \
        fprintf(stderr, "\n"); \
    } \
} while(0)
```

### 8.7.2.3 #define UME\_BLOCK\_PRINT\_ERROR(t,...)

**Value:**

```
do { \
    fprintf(stderr, t, ##__VA_ARGS__); \
    fprintf(stderr, "\n"); \
} while(0)
```

### 8.7.2.4 #define UME\_MALLOC\_RETURN(e, s, r)

**Value:**

```
do { \
    if((e = malloc(s)) == NULL) { \
        return r; \
    } \
} while(0)
```

### 8.7.2.5 #define UME\_SEM\_DESTROY(sem)

**Value:**

```
do { \
    pthread_cond_destroy(&(sem.cond)); \
    pthread_mutex_destroy(&(sem.mtx)); \
} while (0)
```

### 8.7.2.6 #define UME\_SEM\_INIT(sem, len, ret)

**Value:**

```
do { \
    pthread_mutex_init(&(sem.mtx), NULL); \
    pthread_cond_init(&(sem.cond), NULL); \
    sem.max = len; \
    sem.state = len; \
} while(0)
```

### 8.7.2.7 #define UME\_SEM\_POST(sem)

**Value:**

```
do { \
    pthread_mutex_lock(&(sem.mtx)); \
    if(sem.state < sem.max) { sem.state++; } \
    pthread_cond_broadcast(&(sem.cond)); \
    pthread_mutex_unlock(&(sem.mtx)); \
} while(0)
```

#### 8.7.2.8 #define UME\_SEM\_TIMEDWAIT(sem, mstime, ret)

**Value:**

```
do { \
    struct timespec ts; \
    gettimeofday((struct timeval*) &ts, NULL); \
    ts.tv_sec += (mstime/1000); \
    ts.tv_nsec = (ts.tv_nsec*1000) + ((mstime%1000)*1000000); \
    pthread_mutex_lock(&(sem.mtx)); \
    ret = pthread_cond_timedwait(&(sem.cond), &(sem.mtx), &ts); \
    if(ret == 0) { sem.state--; } \
    pthread_mutex_unlock(&(sem.mtx)); \
} while (0)
```

#### 8.7.2.9 #define UME\_SEM\_WAIT(sem)

**Value:**

```
do { \
    pthread_mutex_lock(&(sem.mtx)); \
    while(sem.state == 0) { \
        pthread_cond_wait(&(sem.cond), &(sem.mtx)); } \
    sem.state--; \
    pthread_mutex_unlock(&(sem.mtx)); \
} while(0)
```

### 8.7.3 Function Documentation

#### 8.7.3.1 int ume\_block\_src\_create (ume\_block\_src\_t \*\**srcp*, lbm\_context\_t \**ctx*, lbm\_topic\_t \**topic*, lbm\_src\_topic\_attr\_t \**tattr*, lbm\_src\_cb\_proc*proc*, void \**clientd*, lbm\_event\_queue\_t \**evq*)

**Parameters:**

*srcp* A pointer to a pointer to a UMEBlock source object. Will be filled in by this function to point to a newly created ume\_block\_src\_t object.

*ctx* Pointer to the LBM context object associated with the sender.

***topic*** Pointer to the LBM topic object associated with the destination of messages sent by the source.

***tattr*** Pointer to an LBM topic attribute object. The passed object CANNOT be NULL.

***proc*** Pointer to a function to call when events occur related to the source. If NULL, then events are not delivered to the source.

***clientd*** Pointer to tclient data that is passed when *proc* is called.

***evq*** Optional Event Queue to place events on when they occur. If NULL causes *proc* to be called from context thread.

**Returns:**

0 for Success and -1 for Failure.

### 8.7.3.2 int ume\_block\_src\_delete (ume\_block\_src\_t \* *asrc*)

**Parameters:**

***asrc*** Pointero to an UMEBlock Source object to delete.

**Returns:**

0 for Success and -1 for Failure.

### 8.7.3.3 int ume\_block\_src\_send\_ex (ume\_block\_src\_t \* *asrc*, const char \* *msg*, size\_t *len*, int *flags*, lbm\_src\_send\_ex\_info\_t \* *info*)

**Parameters:**

***asrc*** Pointer to the UMBLOCK source to send from.

***msg*** Pointer to the data to send in this message.

***len*** Length (in bytes) of the data to send in this message.

***info*** Pointer to lbm\_src\_send\_ex\_info\_t options.

**Returns:**

0 for Success and -1 for Failure.

# **Chapter 9**

## **LBM API Page Documentation**

### **9.1 LBMMON Example source code**

Select one of the following for LBMMON example source code.

- [LBMMON LBM transport module](#)
- [LBMMON UDP transport module](#)
- [LBMMON CSV format module](#)
- [LBMMON LBMSNMP transport module](#)

## 9.2 LBMMON LBM transport module

- [lbmmmonrlbm.h](#)
- [lbmmmonrlbm.c](#)

## 9.3 Source code for lbmmmonrlbm.h

```
/** \file lbmmmonrlbm.h
   \brief Ultra Messaging (UM) Monitoring API
   \author David K. Ameiss - Informatica Corporation
   \version $Id: //UMprod/REL_5_3_6/29West/lbm/src/mon/lbm/lbmmmonrlbm.h#2 $

The Ultra Messaging (UM) Monitoring API Description. Included
are types, constants, and functions related to the API. Contents are
subject to change.

All of the documentation and software included in this and any
other Informatica Corporation Ultra Messaging Releases
Copyright (C) Informatica Corporation. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted only as covered by the terms of a
valid software license agreement with Informatica Corporation.

Copyright (C) 2006-2014, Informatica Corporation. All Rights Reserved.

THE SOFTWARE IS PROVIDED "AS IS" AND INFORMATICA DISCLAIMS ALL WARRANTIES
EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF
NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR
PURPOSE. INFORMATICA DOES NOT WARRANT THAT USE OF THE SOFTWARE WILL BE
UNINTERRUPTED OR ERROR-FREE. INFORMATICA SHALL NOT, UNDER ANY CIRCUMSTANCES, BE
LIABLE TO LICENSEE FOR LOST PROFITS, CONSEQUENTIAL, INCIDENTAL, SPECIAL OR
INDIRECT DAMAGES ARISING OUT OF OR RELATED TO THIS AGREEMENT OR THE
TRANSACTIONS CONTEMPLATED HEREUNDER, EVEN IF INFORMATICA HAS BEEN APPRISED OF
THE LIKELIHOOD OF SUCH DAMAGES.

*/
#ifndef LBMMONTRLBM_H
#define LBMMONTRLBM_H

#include <stdlib.h>
#ifdef _WIN32
    #include <winsock2.h>
#endif
#include <lbm/lbmmmon.h>

#if defined(__cplusplus)
extern "C" {
#endif /* __cplusplus */

/*! \brief Return a pointer to the LBMMON_TRANSPORT_LBM module structure.

\return Pointer to LBMMON_TRANSPORT_LBM.

*/
LBMExpDLL const lbmmmon_transport_func_t * lbmmmon_transport_lbm_module(void);

/*! \brief Initialize the LBM transport module to send statistics.

\param TransportClientData A pointer which may be filled in (by this function) with
a pointer to transport-specific client data.
\param TransportOptions The TransportOptions argument originally passed to
lbmmmon_sctl_create().

```

```

        \return Zero if successful, -1 otherwise.
*/
LBMEExpDLL int lbmmon_transport_lbm_initsrc(void * * TransportClientData,
                                              const void * TransportOptions,
                                              TransportOptions * TransportOptions,
                                              void * TransportClientData,
                                              const void * TransportClientData);

/*! \brief Initialize the LBM transport module to receive statistics.

\param TransportClientData A pointer which may be filled in (by this function) with
                           a pointer to transport-specific client data.
\param TransportOptions The TransportOptions argument originally passed to
                        lbmmon_sctl_create().
\return Zero if successful, -1 otherwise.
*/
LBMEExpDLL int lbmmon_transport_lbm_initrcv(void * * TransportClientData,
                                              const void * TransportClientData);

/*! \brief Send a statistics packet.

\param Data The data to be sent.
\param Length The length of the data.
\param TransportClientData A pointer to transport-specific client data.
\return Zero if successful, -1 otherwise.
*/
LBMEExpDLL int lbmmon_transport_lbm_send(const char * Data,
                                          size_t Length,
                                          void * TransportClientData);

/*! \brief Receive statistics packet data.

\param Data A pointer to a buffer to receive the packet data.
\param Length A pointer to a size_t. On entry, it contains the maximum number of bytes
              to receive. On exit, it contains the actual number of bytes received.
\param TimeoutMS Maximum timeout in milliseconds. If no data is available within
                  the timeout value, return.
\param TransportClientData A pointer to transport-specific client data.
\return Zero if successful, -1 otherwise.
*/
LBMEExpDLL int lbmmon_transport_lbm_receive(char * Data,
                                             size_t * Length,
                                             unsigned int TimeoutMS,
                                             void * TransportClientData);

/*! \brief Finish LBM transport module source processing.

\param TransportClientData A pointer to transport-specific client data.
\return Zero if successful, -1 otherwise.
*/
LBMEExpDLL int lbmmon_transport_lbm_src_finish(void * TransportClientData);

/*! \brief Finish LBM transport module receiver processing.

\param TransportClientData A pointer to transport-specific client data.
\return Zero if successful, -1 otherwise.
*/
LBMEExpDLL int lbmmon_transport_lbm_rcv_finish(void * TransportClientData);

/*! \brief Return a messages describing the last error encountered.

```

```
\return A string containing a description of the last error encountered by the module.  
*/  
LBMExpDLL const char * lbmmmon_transport_lbm_errmsg(void);  
  
#if defined(__cplusplus)  
}  
#endif /* __cplusplus */  
  
#endif
```

## 9.4 Source code for lbmmmonrlbm.c

```
/*
All of the documentation and software included in this and any
other Informatica Corporation Ultra Messaging Releases
Copyright (C) Informatica Corporation. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted only as covered by the terms of a
valid software license agreement with Informatica Corporation.

Copyright (C) 2004-2014, Informatica Corporation. All Rights Reserved.

THE SOFTWARE IS PROVIDED "AS IS" AND INFORMATICA DISCLAIMS ALL WARRANTIES
EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF
NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR
PURPOSE. INFORMATICA DOES NOT WARRANT THAT USE OF THE SOFTWARE WILL BE
UNINTERRUPTED OR ERROR-FREE. INFORMATICA SHALL NOT, UNDER ANY CIRCUMSTANCES, BE
LIABLE TO LICENSEE FOR LOST PROFITS, CONSEQUENTIAL, INCIDENTAL, SPECIAL OR
INDIRECT DAMAGES ARISING OUT OF OR RELATED TO THIS AGREEMENT OR THE
TRANSACTIONS CONTEMPLATED HEREUNDER, EVEN IF INFORMATICA HAS BEEN APPRISED OF
THE LIKELIHOOD OF SUCH DAMAGES.

*/
#ifndef __VOS__
#define _POSIX_C_SOURCE 200112L
#include <sys/time.h>
#endif

#include <stdio.h>
#include <time.h>
#include <string.h>
#ifdef _WIN32
    #define strcasecmp strcmp
    #define sprintf _snprintf
#else
    #include "config.h"
    #include <unistd.h>
    #if defined(__TANDEM)
        #if defined(HAVE_TANDEM_SPT)
            #include <ktdmtyp.h>
            #include <spthread.h>
        #else
            #include <pthread.h>
        #endif
    #else
        #include <pthread.h>
    #endif
    #include <strings.h>
#endif
#include <lbm/lbmmmon.h>
#include <lbm/lbmmmonrlbm.h>
#include <lbm/lbmaux.h>

/*
    Package all of the needed function pointers for this module into a
```

```

        lbmmmon_transport_func_t structure.

*/
static const lbmmmon_transport_func_t LBMMON_TRANSPORT_LBM =
{
    lbmmmon_transport_lbm_initsrc,
    lbmmmon_transport_lbm_initrcv,
    lbmmmon_transport_lbm_send,
    lbmmmon_transport_lbm_receive,
    lbmmmon_transport_lbm_src_finish,
    lbmmmon_transport_lbm_rcv_finish,
    lbmmmon_transport_lbm_errmsg
};

/*
     For a statistics source, one of these gets returned as the TransportClientData.
*/
typedef struct
{
    /* LBM context attributes */
    lbm_context_attr_t * mContextAttributes;
    /* LBM context created to send a statistics packet */
    lbm_context_t * mContext;
    /* LBM topic attributes */
    lbm_src_topic_attr_t * mTopicAttributes;
    /* LBM source created to send a statistics packet */
    lbm_src_t * mSource;
    /* LBM topic */
    lbm_topic_t * mTopic;
} lbmmmon_transport_lbm_src_t;

/*
     A queue of incoming statistics packets is maintained. This describes each
     entry in the queue.
*/
struct lbmmmon_transport_lbm_rcv_node_t_stct
{
    /* Pointer to the LBM message */
    lbm_msg_t * mMessage;
    /* Number of bytes of the message returned to caller */
    size_t mUsedBytes;
    /* Next entry in the queue */
    struct lbmmmon_transport_lbm_rcv_node_t_stct * mNext;
};
typedef struct lbmmmon_transport_lbm_rcv_node_t_stct lbmmmon_transport_lbm_rcv_node_t;

/*
     For a statistics receiver, one of these gets returned as the TransportClientData.
*/
typedef struct
{
    /* Flag to indicate lock has been created */
    unsigned int mLockCreated;
    /* Lock to prevent access by multiple threads */
#ifndef _WIN32
    CRITICAL_SECTION mLock;
#else
    pthread_mutex_t mLock;

```

```

#endif

/* LBM context attributes */
lbm_context_attr_t * mContextAttributes;
/* LBM context used to receive packets */
lbm_context_t * mContext;
/* LBM receiver used to receive packets */
lbm_rcv_t * mReceiver;
/* Topic attributes */
lbm_rcv_topic_attr_t * mTopicAttributes;
/* Topic */
lbm_topic_t * mTopic;
/* Wildcard receiver attributes */
lbm_wildcard_rcv_attr_t * mWildcardReceiverAttributes;
/* If we're using a wildcard receiver... */
lbm_wildcard_rcv_t * mWildcardReceiver;
/* Head of the message queue */
lbmmmon_transport_lbm_rcv_node_t * mHead;
/* Tail of the message queue */
lbmmmon_transport_lbm_rcv_node_t * mTail;
} lbmmmon_transport_lbm_rcv_t;

static void      src_cleanup(lbmmmon_transport_lbm_src_t * Data);
static void      rcv_cleanup(lbmmmon_transport_lbm_rcv_t * Data);
static int receive_callback(lbm_rcv_t * Receiver, lbm_msg_t * Message, void * ClientData);
static void lock_receiver(lbmmmon_transport_lbm_rcv_t * Receiver);
static void unlock_receiver(lbmmmon_transport_lbm_rcv_t * Receiver);
static int scope_is_valid(const char * Scope);

#define DEFAULT_CONTEXT_NAME "29west_statistics_context"
#define DEFAULT_TOPIC "/29west/statistics"

static char ErrorString[1024];

typedef struct
{
    const char * option;
    const char * value;
} option_entry_t;

static option_entry_t SourceContextOption[] =
{
    /* Force embedded mode for simplicity. */
    { "operational_mode", "embedded" },
    /* Disable monitoring for this context. */
    { "monitor_interval", "0" },
    /* We don't need request/response, so don't use up ports. */
    { "request_tcp_bind_request_port", "0" },
    /* We don't need MIM, so disable MIM receiver. */
    { "mim_incoming_address", "0.0.0.0" },
    /* No need to cache topics. */
    { "resolver_cache", "0" },
    /* End of list. */
    { NULL, NULL }
};

static option_entry_t ReceiverContextOption[] =
{

```

```

/* Force embedded mode for simplicity. */
{ "operational_mode", "embedded" },
/* Disable monitoring for this context. */
{ "monitor_interval", "0" },
/* We don't need request/response, so don't use up ports. */
{ "request_tcp_bind_request_port", "0" },
/* We don't need MIM, so disable MIM receiver. */
{ "mim_incoming_address", "0.0.0.0" },
/* No need to cache topics. */
{ "resolver_cache", "0" },
/* End of list. */
{ NULL, NULL }
};

static option_entry_t SourceTopicOption[] =
{
    /* Minimize memory used for LBT-RU retransmissions. */
    { "transport_lbtru_transmission_window_size", "500000" },
    /* Minimize memory used for LBT-RM retransmissions. */
    { "transport_lbtrm_transmission_window_size", "500000" },
    /* End of list. */
    { NULL, NULL }
};

static option_entry_t ReceiverTopicOption[] =
{
    /* End of list. */
    { NULL, NULL }
};

static option_entry_t WildcardReceiverOption[] =
{
    /* End of list. */
    { NULL, NULL }
};

const lbmon_transport_func_t *
lbmon_transport_lbm_module(void)
{
    return (&LBMMON_TRANSPORT_LBM);
}

int
lbmon_transport_lbm_initsrc(void * * TransportClientData, const void * TransportOptions)
{
    lbmon_transport_lbm_src_t * data;
    int rc;
    const char * ptr = (const char *) TransportOptions;
    char key[512];
    char value[512];
    char config_file[512];
    char topic[512];
    char scope[512];
    char option[512];
    option_entry_t * entry;

    memset(ErrorString, 0, sizeof(ErrorString));
}

```

```

data = malloc(sizeof(lbmmon_transport_lbm_src_t));
data->mContextAttributes = NULL;
data->mContext = NULL;
data->mTopicAttributes = NULL;
data->mSource = NULL;
data->mTopic = NULL;

/* Process any options */
memset(config_file, 0, sizeof(config_file));
strncpy(topic, DEFAULT_TOPIC, sizeof(topic));
while ((ptr = lbmmon_next_key_value_pair(ptr, key, value, sizeof(key), value, sizeof(value))) != NULL)
{
    if (strcasecmp(key, "config") == 0)
    {
        strncpy(config_file, value, sizeof(config_file));
    }
    else if (strcasecmp(key, "topic") == 0)
    {
        strncpy(topic, value, sizeof(topic));
    }
}

/* Initialize the context attributes */
rc = lbm_context_attr_create_default(&(data->mContextAttributes));
if (rc == LBM_FAILURE)
{
    sprintf(ErrorString,
            sizeof(ErrorString),
            "lbm_context_attr_init() failed, %s",
            lbm_errmsg());
    return (rc);
}
/* Set the default context name */
rc = lbm_context_attr_str_setopt(data->mContextAttributes, "context_name", DEFAULT_CONTEXT_NAME);
if (rc == LBM_FAILURE)
{
    sprintf(ErrorString,
            sizeof(ErrorString),
            "lbm_context_attr_str_setopt() failed, %s",
            lbm_errmsg());
    return (rc);
}
if (config_file[0] != '\0')
{
    /* A config file was passed as an option. Use it to populate the context attributes */
    rc = lbmaux_context_attr_setopt_from_file(data->mContextAttributes, config_file);
    if (rc != 0)
    {
        sprintf(ErrorString,
                sizeof(ErrorString),
                "lbmaux_context_attr_setopt_from_file() failed, %s",
                lbm_errmsg());
        src_cleanup(data);
        return (-1);
    }
}
/* Go back through the options, looking for any specific context options. */

```

```

ptr = (const char *) TransportOptions;
while ((ptr = lbmon_next_key_value_pair(ptr, key, sizeof(key), value, sizeof(value))) != NULL)
{
    if (sscanf(key, "%[a-zA-Z_]%[a-zA-Z_]", scope, option) != 2)
    {
        continue;
    }
    if (scope_is_valid(scope) == -1)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),
                 "invalid option scope [%s]",
                 scope);
        src_cleanup(data);
        return (-1);
    }
    if (strcasecmp(scope, "context") == 0)
    {
        rc = lbm_context_attr_str_setopt(data->mContextAttributes, option, value);
        if (rc == LBM_FAILURE)
        {
            snprintf(ErrorString,
                     sizeof(ErrorString),
                     "invalid option [context %s %s], %s",
                     option,
                     value,
                     lbm_errmsg());
            src_cleanup(data);
            return (rc);
        }
    }
}

entry = &SourceContextOption[0];
while (entry->option != NULL)
{
    rc = lbm_context_attr_str_setopt(data->mContextAttributes, entry->option, entry->value);
    if (rc == LBM_FAILURE)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),
                 "error setting option [context %s %s], %s",
                 entry->option,
                 entry->value,
                 lbm_errmsg());
        src_cleanup(data);
        return (rc);
    }
    entry++;
}

/* Create the context */
rc = lbm_context_create(&(data->mContext), data->mContextAttributes, NULL, NULL);
if (rc == LBM_FAILURE)
{
    snprintf(ErrorString,
             sizeof(ErrorString),

```

```

                "lbt_context_create() failed, %s",
                lbt_errmsg());
        src_cleanup(data);
        return (rc);
    }

/* Initialize the source topic attributes */
rc = lbt_src_topic_attr_create_default(&(data->mTopicAttributes));
if (rc == LBT_FAILURE)
{
    snprintf(ErrorString,
            sizeof(ErrorString),
            "lbt_src_topic_attr_create_default() failed, %s",
            lbt_errmsg());
    src_cleanup(data);
    return (rc);
}
entry = &SourceTopicOption[0];
while (entry->option != NULL)
{
    rc = lbt_src_topic_attr_str_setopt(data->mTopicAttributes, entry->option, entry
    if (rc == LBT_FAILURE)
    {
        snprintf(ErrorString,
                sizeof(ErrorString),
                "error setting option [source %s %s], %s",
                entry->option,
                entry->value,
                lbt_errmsg());
        src_cleanup(data);
        return (rc);
    }
    entry++;
}
if (config_file[0] != '\0')
{
    /* A config file was passed as an option. Use it to populate the source topic attributes */
    rc = lbmaux_src_topic_attr_setopt_from_file(data->mTopicAttributes, config_file);
    if (rc != 0)
    {
        snprintf(ErrorString,
                sizeof(ErrorString),
                "lbmaux_src_topic_attr_setopt_from_file() failed, %s",
                lbt_errmsg());
        src_cleanup(data);
        return (-1);
    }
}
/* Go back through the options, looking for any specific source options. */
ptr = (const char *) TransportOptions;
while ((ptr = lbmon_next_key_value_pair(ptr, key, sizeof(key), value, sizeof(value))) != NULL)
{
    if (sscanf(key, "%[a-zA-Z_]%[a-zA-Z_]", scope, option) != 2)
    {
        continue;
    }
    if (strcasecmp(scope, "source") == 0)

```

```

{
    rc = lbm_src_topic_attr_str_setopt(data->mTopicAttributes, option, value);
    if (rc == LBM_FAILURE)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),
                 "invalid option [source %s %s], %s",
                 option,
                 value,
                 lbm_errmsg());
        src_cleanup(data);
        return (rc);
    }
}

/* Create the topic */
rc = lbm_src_topic_alloc(&(data->mTopic), data->mContext, topic, data->mTopicAttributes);
if (rc == LBM_FAILURE)
{
    snprintf(ErrorString,
             sizeof(ErrorString),
             "lbm_src_topic_alloc() failed, %s",
             lbm_errmsg());
    src_cleanup(data);
    return (rc);
}

/* Create the source */
rc = lbm_src_create(&(data->mSource), data->mContext, data->mTopic, NULL, NULL, NULL);
if (rc == LBM_FAILURE)
{
    snprintf(ErrorString,
             sizeof(ErrorString),
             "lbm_src_create() failed, %s",
             lbm_errmsg());
    src_cleanup(data);
    return (rc);
}

/* Pass back the lbmmmon_transport_lbm_src_t created */
*TransportClientData = data;
return (0);
}

/*
This function is called upon receipt of an LBM message (when operating as
a statistics receiver).
*/
int
receive_callback(lbm_rcv_t * Receiver, lbm_msg_t * Message, void * ClientData)
{
    lbmmmon_transport_lbm_rcv_t * rcv = (lbmmmon_transport_lbm_rcv_t *) ClientData;
    lbmmmon_transport_lbm_rcv_node_t * node;

    if (Message->type == LBM_MSG_DATA)
    {

```

```

/* A data message. We want to enqueue it for processing. */
lock_receiver(rcv);
node = malloc(sizeof(lbmmon_transport_lbm_rcv_node_t));
/*
   Since we hold onto the message until it is actually processed,
   let LBM know about it.
*/
lbm_msg_retain(Message);
node->mMessage = Message;
node->mUsedBytes = 0; /* No data returned as yet */

/* Link the message onto the queue */
node->mNext = NULL;
if (rcv->mTail != NULL)
{
    rcv->mTail->mNext = node;
}
else
{
    rcv->mHead = node;
}
rcv->mTail = node;
unlock_receiver(rcv);
}
return (0);
}

int
lbmmon_transport_lbm_initrcv(void * * TransportClientData, const void * TransportOptions)
{
    lbmmon_transport_lbm_rcv_t * data;
    int rc;
    const char * ptr = (const char *) TransportOptions;
    char key[512];
    char value[512];
    char config_file[512];
    char topic[512];
    char wildcard_topic[512];
    char scope[512];
    char option[512];
    option_entry_t * entry;

    memset(ErrorString, 0, sizeof(ErrorString));
    data = malloc(sizeof(lbmmon_transport_lbm_rcv_t));

    data->mLockCreated = 0;
    data->mContextAttributes = NULL;
    data->mContext = NULL;
    data->mReceiver = NULL;
    data->mTopicAttributes = NULL;
    data->mTopic = NULL;
    data->mWildcardReceiverAttributes = NULL;
    data->mWildcardReceiver = NULL;
    data->mHead = NULL;
    data->mTail = NULL;

    /* Process any options */

```

```

        memset(config_file, 0, sizeof(config_file));
        strncpy(topic, DEFAULT_TOPIC, sizeof(topic));
        memset(wildcard_topic, 0, sizeof(wildcard_topic));
        while ((ptr = lbmmmon_next_key_value_pair(ptr, key, sizeof(key), value, sizeof(value))) != NULL)
        {
            if (strcasecmp(key, "config") == 0)
            {
                strncpy(config_file, value, sizeof(config_file));
            }
            else if (strcasecmp(key, "topic") == 0)
            {
                strncpy(topic, value, sizeof(topic));
            }
            else if (strcasecmp(key, "wctopic") == 0)
            {
                strncpy(wildcard_topic, value, sizeof(wildcard_topic));
            }
        }

        /* Initialize the context attributes */
        rc = lbm_context_attr_create_default(&(data->mContextAttributes));
        if (rc == LBM_FAILURE)
        {
            snprintf(ErrorString,
                     sizeof(ErrorString),
                     "lbm_context_attr_init() failed, %s",
                     lbm_errmsg());
            rcv_cleanup(data);
            return (rc);
        }
        /* Set the default context name */
        rc = lbm_context_attr_str_setopt(data->mContextAttributes, "context_name", DEFAULT_CONTEXT_NAME);
        if (rc == LBM_FAILURE)
        {
            snprintf(ErrorString,
                     sizeof(ErrorString),
                     "lbm_context_attr_str_setopt() failed, %s",
                     lbm_errmsg());
            return (rc);
        }
        if (config_file[0] != '\0')
        {
            /* A config file was passed as an option. Use it to populate the context attributes. */
            rc = lbmaux_context_attr_setopt_from_file(data->mContextAttributes, config_file);
            if (rc != 0)
            {
                snprintf(ErrorString,
                         sizeof(ErrorString),
                         "lbmaux_context_attr_setopt_from_file() failed, %s",
                         lbm_errmsg());
                rcv_cleanup(data);
                return (-1);
            }
        }
        /* Go back through the options, looking for any specific context options. */
        ptr = (const char *) TransportOptions;
        while ((ptr = lbmmmon_next_key_value_pair(ptr, key, sizeof(key), value, sizeof(value))) != NULL)

```

```

{
    if (sscanf(key, "%[a-zA-Z_]%[a-zA-Z_]", scope, option) != 2)
    {
        continue;
    }
    if (scope_is_valid(scope) == -1)
    {
        snprintf(ErrorString,
                  sizeof(ErrorString),
                  "invalid option scope [%s]",
                  scope);
        rcv_cleanup(data);
        return (-1);
    }
    if (strcasecmp(scope, "context") == 0)
    {
        rc = lbm_context_attr_str_setopt(data->mContextAttributes, option, value);
        if (rc == LBM_FAILURE)
        {
            snprintf(ErrorString,
                  sizeof(ErrorString),
                  "invalid option [context %s %s], %s",
                  option,
                  value,
                  lbm_errmsg());
            rcv_cleanup(data);
            return (rc);
        }
    }
}

entry = &ReceiverContextOption[0];
while (entry->option != NULL)
{
    rc = lbm_context_attr_str_setopt(data->mContextAttributes, entry->option, entry->value);
    if (rc == LBM_FAILURE)
    {
        snprintf(ErrorString,
                  sizeof(ErrorString),
                  "error setting option [context %s %s], %s",
                  entry->option,
                  entry->value,
                  lbm_errmsg());
        rcv_cleanup(data);
        return (rc);
    }
    entry++;
}

/* Create the context */
rc = lbm_context_create(&(data->mContext), data->mContextAttributes, NULL, NULL);
if (rc == LBM_FAILURE)
{
    snprintf(ErrorString,
                  sizeof(ErrorString),
                  "lbm_context_create() failed, %s",
                  lbm_errmsg());
}

```

```

        rcv_cleanup(data);
        return (rc);
    }

/* If a wildcard topic was specified, initialize the wildcard receiver attributes. */
if (wildcard_topic[0] != '\0')
{
    rc = lbm_wildcard_rcv_attr_create_default(&(data->mWildcardReceiverAttributes));
    if (rc == LBM_FAILURE)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),
                 "lbm_wildcard_rcv_attr_init() failed, %s",
                 lbm_errmsg());
        rcv_cleanup(data);
        return (rc);
    }
    if (config_file[0] != '\0')
    {
        /* A config file was passed as an option. Use it to populate the wildcard receiver
        rc = lbmaux_wildcard_rcv_attr_setopt_from_file(data->mWildcardReceiverAttributes,
        if (rc == LBM_FAILURE)
        {
            snprintf(ErrorString,
                     sizeof(ErrorString),
                     "lbmaux_wildcard_rcv_attr_setopt_from_file() failed, %s",
                     lbm_errmsg());
            rcv_cleanup(data);
            return (-1);
        }
    }
    /* Go back through the options, looking for any specific wildcard receiver options. */
    ptr = (const char *) TransportOptions;
    while ((ptr = lbmmmon_next_key_value_pair(ptr, key, sizeof(key), value, sizeof(value))) !=
    {
        if (sscanf(key, "%[a-zA-Z_]%[a-zA-Z_]", scope, option) != 2)
        {
            continue;
        }
        if (strcasecmp(scope, "wildcard_receiver") == 0)
        {
            rc = lbm_wildcard_rcv_attr_str_setopt(data->mWildcardReceiverAttributes,
            if (rc == LBM_FAILURE)
            {
                snprintf(ErrorString,
                         sizeof(ErrorString),
                         "invalid option [wildcard_receiver %s %s], %s",
                         option,
                         value,
                         lbm_errmsg());
                rcv_cleanup(data);
                return (rc);
            }
        }
    }
    entry = &WildcardReceiverOption[0];
    while (entry->option != NULL)

```

```

{
    rc = lbm_wildcard_rcv_attr_str_setopt(data->mWildcardReceiverAttributes,
    if (rc == LBM_FAILURE)
    {
        snprintf(ErrorString,
                sizeof(ErrorString),
                "error setting option [wildcard_receiver %s %s",
                entry->option,
                entry->value,
                lbm_errmsg());
        rcv_cleanup(data);
        return (rc);
    }
    entry++;
}
}

/* Initialize and set the receiver topic attributes. */
rc = lbm_rcv_topic_attr_create_default(&(data->mTopicAttributes));
if (rc == LBM_FAILURE)
{
    snprintf(ErrorString,
            sizeof(ErrorString),
            "lbm_rcv_topic_attr_init() failed, %s",
            lbm_errmsg());
    rcv_cleanup(data);
    return (rc);
}
if (config_file[0] != '\0')
{
    /* A config file was passed as an option. Use it to populate the receiver topic
    rc = lbmaux_rcv_topic_attr_setopt_from_file(data->mTopicAttributes, config_file);
    if (rc == LBM_FAILURE)
    {
        snprintf(ErrorString,
                sizeof(ErrorString),
                "lbmaux_rcv_topic_attr_setopt_from_file() failed, %s",
                lbm_errmsg());
        rcv_cleanup(data);
        return (-1);
    }
}
/* Go back through the options, looking for any specific receiver options. */
ptr = (const char *) TransportOptions;
while ((ptr = lbmmon_next_key_value_pair(ptr, key, sizeof(key), value, sizeof(value))) != NULL)
{
    if (sscanf(key, "%[a-zA-Z_]|%[a-zA-Z_]", scope, option) != 2)
    {
        continue;
    }
    if (strcasecmp(scope, "receiver") == 0)
    {
        rc = lbm_rcv_topic_attr_str_setopt(data->mTopicAttributes, option, value);
        if (rc == LBM_FAILURE)
        {
            snprintf(ErrorString,
                    sizeof(ErrorString),
                    "error setting option [receiver %s %s",
                    option,
                    value,
                    lbm_errmsg());
            rcv_cleanup(data);
            return (-1);
        }
    }
}

```

```

        "invalid option [receiver %s %s], %s",
        option,
        value,
        lbm_errmsg());
    rcv_cleanup(data);
    return (rc);
}
}

entry = &ReceiverTopicOption[0];
while (entry->option != NULL)
{
    rc = lbm_rcv_topic_attr_str_setopt(data->mTopicAttributes, entry->option, entry->value);
    if (rc == LBM_FAILURE)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),
                 "error setting option [receiver %s %s], %s",
                 entry->option,
                 entry->value,
                 lbm_errmsg());
        rcv_cleanup(data);
        return (rc);
    }
    entry++;
}

/* For a non-wildcard topic, lookup the topic. */
if (wildcard_topic[0] == '\0')
{
    rc = lbm_rcv_topic_lookup(&(data->mTopic), data->mContext, topic, data->mTopicAttributes);
    if (rc == LBM_FAILURE)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),
                 "lbm_rcv_topic_lookup() failed, %s",
                 lbm_errmsg());
        rcv_cleanup(data);
        return (rc);
    }
}

#endif _WIN32
    InitializeCriticalSection(&(data->mLock));
#else
    pthread_mutex_init(&(data->mLock), NULL);
#endif
data->mLockCreated = 1;
lock_receiver(data);
if (wildcard_topic[0] != '\0')
{
    /* Wildcard topic, create a wildcard receiver */
    rc = lbm_wildcard_rcv_create(&(data->mWildcardReceiver),
                                 data->mContext,
                                 wildcard_topic,
                                 data->mTopicAttributes,
                                 data->mWildcardReceiverAttributes);
}

```

```

        receive_callback,
        data,
        NULL);
    }
    else
    {
        /* Non-wildcard topic, create a normal receiver */
        rc = lbm_rcv_create(&(data->mReceiver),
                            data->mContext,
                            data->mTopic,
                            receive_callback,
                            data,
                            NULL);
    }
    if (rc == LBM_FAILURE)
    {
        sprintf(ErrorString,
                sizeof(ErrorString),
                "lbm_wildcard_rcv_create()/lbm_rcv_create() failed, %s",
                lbm_errmsg());
        unlock_receiver(data);
        rcv_cleanup(data);
        return (rc);
    }

    /* Pass back the lbmmmon_transport_lbm_rcv_t created */
    *TransportClientData = data;
    unlock_receiver(data);
    return (0);
}

int
lbmmmon_transport_lbm_send(const char * Data, size_t Length, void * TransportClientData)
{
    lbmmmon_transport_lbm_src_t * src;
    int rc;

    if ((Data == NULL) || (TransportClientData == NULL))
    {
        strncpy(ErrorString, "Invalid argument", sizeof(ErrorString));
        return (-1);
    }
    src = (lbmmmon_transport_lbm_src_t *) TransportClientData;
    rc = lbm_src_send(src->mSource, Data, Length, 0);
    if (rc == LBM_FAILURE)
    {
        sprintf(ErrorString,
                sizeof(ErrorString),
                "lbm_src_send() failed, %s",
                lbm_errmsg());
    }
    return (rc);
}

int
lbmmmon_transport_lbm_receive(char * Data, size_t * Length, unsigned int TimeoutMS, void * Trans
}

```

```

lbmmmon_transport_lbm_rcv_t * rcv = (lbmmmon_transport_lbm_rcv_t *) TransportClientData;
lbmmmon_transport_lbm_rcv_node_t * node;
int rc = 0;
size_t length_remaining;
#if defined(_WIN32)
#elif defined(__TANDEM)
    unsigned int sleep_sec;
    unsigned int sleep_usec;
#else
    struct timespec ivl;
#endif

if ((Data == NULL) || (Length == NULL) || (TransportClientData == NULL))
{
    strncpy(ErrorString, "Invalid argument", sizeof(ErrorString));
    return (-1);
}
if (*Length == 0)
{
    return (0);
}
lock_receiver(rcv);
if (rcv->mHead != NULL)
{
    /* Queue is non-empty. Pull the first message from the queue. */
    node = rcv->mHead;
    length_remaining = node->mMessage->len - node->mUsedBytes;
    if (*Length >= length_remaining)
    {
        /* We can transfer the rest of the message */
        memcpy(Data, node->mMessage->data + node->mUsedBytes, length_remaining);
        *Length = length_remaining;
        rc = 0;
        /* We're done with the LBM message, so let LBM know. */
        lbm_msg_delete(node->mMessage);
        /* Unlink the node from the queue */
        rcv->mHead = node->mNext;
        if (rcv->mHead == NULL)
        {
            rcv->mTail = NULL;
        }
        free(node);
    }
    else
    {
        /* Can only transfer part of the message */
        memcpy(Data, node->mMessage->data + node->mUsedBytes, *Length);
        node->mUsedBytes -= *Length;
        rc = 0;
    }
    unlock_receiver(rcv);
}
else
{
    unlock_receiver(rcv);
    /* Sleep for wait time */
#define NANOSECONDS_PER_SECOND 1000000000

```

```

#define MICROSECONDS_PER_SECOND 1000000
#define MILLISECONDS_PER_SECOND 1000
#define NANOSECONDS_PER_MILLISECOND (NANOSECONDS_PER_SECOND / MILLISECONDS_PER_SECOND)
#define MICROSECONDS_PER_MILLISECOND (MICROSECONDS_PER_SECOND / MILLISECONDS_PER_SECOND)
#if defined(_WIN32)
    Sleep(TimeoutMS);
#elif defined(__TANDEM)
    sleep_sec = TimeoutMS / MILLISECONDS_PER_SECOND;
    sleep_usec = (TimeoutMS % MILLISECONDS_PER_SECOND) * MICROSECONDS_PER_MILLISECOND;
    if (sleep_usec > 0)
    {
        usleep(sleep_usec);
    }
    if (sleep_sec > 0)
    {
        sleep(sleep_sec);
    }
#else
    ivl.tv_sec = TimeoutMS / MILLISECONDS_PER_SECOND;
    ivl.tv_nsec = (TimeoutMS % MILLISECONDS_PER_SECOND) * NANOSECONDS_PER_MILLISECOND;
    nanosleep(&ivl, NULL);
#endif
    rc = 1;
}
return (rc);
}

void
src_cleanup(lbmmmon_transport_lbm_src_t * Data)
{
    if (Data->mSource != NULL)
    {
        lbm_src_delete(Data->mSource);
        Data->mSource = NULL;
    }
    Data->mTopic = NULL;
    if (Data->mTopicAttributes != NULL)
    {
        lbm_src_topic_attr_delete(Data->mTopicAttributes);
        Data->mTopicAttributes = NULL;
    }
    if (Data->mContext != NULL)
    {
        lbm_context_delete(Data->mContext);
        Data->mContext = NULL;
    }
    if (Data->mContextAttributes != NULL)
    {
        lbm_context_attr_delete(Data->mContextAttributes);
        Data->mContextAttributes = NULL;
    }
    free(Data);
}

int
lbmmmon_transport_lbm_src_finish(void * TransportClientData)
{

```

```
lbmmmon_transport_lbm_src_t * src;

if (TransportClientData == NULL)
{
    strncpy(ErrorString, "Invalid argument", sizeof(ErrorString));
    return (-1);
}
src = (lbmmmon_transport_lbm_src_t *) TransportClientData;
src_cleanup(src);
return (0);
}

void
rcv_cleanup(lbmmmon_transport_lbm_rcv_t * Data)
{
    lbmmmon_transport_lbm_rcv_node_t * node;
    lbmmmon_transport_lbm_rcv_node_t * next;

    /* Stop the receiver to prevent any more incoming messages */
    if (Data->mWildcardReceiver != NULL)
    {
        lbm_wildcard_rcv_delete(Data->mWildcardReceiver);
        Data->mWildcardReceiver = NULL;
    }
    if (Data->mWildcardReceiverAttributes != NULL)
    {
        lbm_wildcard_rcv_attr_delete(Data->mWildcardReceiverAttributes);
        Data->mWildcardReceiverAttributes = NULL;
    }
    if (Data->mReceiver != NULL)
    {
        lbm_rcv_delete(Data->mReceiver);
        Data->mReceiver = NULL;
    }
    if (Data->mTopicAttributes != NULL)
    {
        lbm_rcv_topic_attr_delete(Data->mTopicAttributes);
        Data->mTopicAttributes = NULL;
    }
    Data->mTopic = NULL;

    /* Lock the receiver */
    if (Data->mLockCreated != 0)
    {
        lock_receiver(Data);
    }

    /* Delete the context to really make sure no more messages come in */
    if (Data->mContext != NULL)
    {
        lbm_context_delete(Data->mContext);
        Data->mContext = NULL;
    }
    if (Data->mContextAttributes != NULL)
    {
        lbm_context_attr_delete(Data->mContextAttributes);
        Data->mContextAttributes = NULL;
```

```

    }

    /* Clean out the queue */
    node = Data->mHead;
    while (node != NULL)
    {
        /* Let LBM know we're done with the message */
        lbm_msg_delete(node->mMessage);
        next = node->mNext;
        free(node);
        node = next;
    }

    if (Data->mLockCreated)
    {
        unlock_receiver(Data);
#ifndef _WIN32
        DeleteCriticalSection(&(Data->mLock));
#else
        pthread_mutex_destroy(&(Data->mLock));
#endif
    }

    free(Data);
}

int
lbmmmon_transport_lbm_rcv_finish(void * TransportClientData)
{
    lbmmmon_transport_lbm_rcv_t * rcv;

    if (TransportClientData == NULL)
    {
        strncpy(ErrorString, "Invalid argument", sizeof(ErrorString));
        return (-1);
    }
    rcv = (lbmmmon_transport_lbm_rcv_t *) TransportClientData;
    rcv_cleanup(rcv);
    return (0);
}

void
lock_receiver(lbmmmon_transport_lbm_rcv_t * Receiver)
{
#ifndef _WIN32
    EnterCriticalSection(&(Receiver->mLock));
#else
    pthread_mutex_lock(&(Receiver->mLock));
#endif
}

void
unlock_receiver(lbmmmon_transport_lbm_rcv_t * Receiver)
{
#ifndef _WIN32
    LeaveCriticalSection(&(Receiver->mLock));
#else

```

```
    pthread_mutex_unlock(&(Receiver->mLock));  
#endif  
}  
  
const char *  
lbmmmon_transport_lbm_errmsg(void)  
{  
    return (ErrorString);  
}  
  
int  
scope_is_valid(const char * Scope)  
{  
    if (strcasecmp(Scope, "context") == 0)  
    {  
        return (0);  
    }  
    if (strcasecmp(Scope, "source") == 0)  
    {  
        return (0);  
    }  
    if (strcasecmp(Scope, "receiver") == 0)  
    {  
        return (0);  
    }  
    if (strcasecmp(Scope, "event_queue") == 0)  
    {  
        return (0);  
    }  
    return (-1);  
}
```

## 9.5 LBMMON UDP transport module

- [lbmmmontrudp.h](#)
- [lbmmmontrudp.c](#)

## 9.6 Source code for lbmontrudp.h

```
/** \file lbmontrudp.h
   \brief Ultra Messaging (UM) Monitoring API
   \author David K. Ameiss - Informatica Corporation
   \version $Id: //UMprod/REL_5_3_6/29West/lbm/src/mon/lbm/lbmontrudp.h#2 $

The Ultra Messaging (UM) Monitoring API Description. Included
are types, constants, and functions related to the API. Contents are
subject to change.

All of the documentation and software included in this and any
other Informatica Corporation Ultra Messaging Releases
Copyright (C) Informatica Corporation. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted only as covered by the terms of a
valid software license agreement with Informatica Corporation.

Copyright (C) 2006-2014, Informatica Corporation. All Rights Reserved.

THE SOFTWARE IS PROVIDED "AS IS" AND INFORMATICA DISCLAIMS ALL WARRANTIES
EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF
NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR
PURPOSE. INFORMATICA DOES NOT WARRANT THAT USE OF THE SOFTWARE WILL BE
UNINTERRUPTED OR ERROR-FREE. INFORMATICA SHALL NOT, UNDER ANY CIRCUMSTANCES, BE
LIABLE TO LICENSEE FOR LOST PROFITS, CONSEQUENTIAL, INCIDENTAL, SPECIAL OR
INDIRECT DAMAGES ARISING OUT OF OR RELATED TO THIS AGREEMENT OR THE
TRANSACTIONS CONTEMPLATED HEREUNDER, EVEN IF INFORMATICA HAS BEEN APPRISED OF
THE LIKELIHOOD OF SUCH DAMAGES.

*/
#ifndef LBMMONTRUDP_H
#define LBMMONTRUDP_H

#include <stdlib.h>
#ifdef _WIN32
    #include <winsock2.h>
#endif
#include <lbt/lbmon.h>

#if defined(__cplusplus)
extern "C" {
#endif /* __cplusplus */

/*! \brief Return a pointer to the LBMMON_TRANSPORT_UDP module structure.

\return Pointer to LBMMON_TRANSPORT_UDP.

*/
LBMEExpDLL const lbmon_transport_func_t * lbmon_udp_module(void);

/*!
\brief Initialize the UDP transport module to send statistics.

\param TransportClientData A pointer which may be filled in (by this function) with
a pointer to transport-specific client data.
\param TransportOptions The TransportOptions argument originally passed to
lbmon_sctl_create().

```

```

        \return Zero if successful, -1 otherwise.
*/
LBMEExpDLL int lbmmon_transport_udp_initsrc(void * * TransportClientData,
                                              const void * Transpor
                                         tClientData,
                                         TransportOptions Options,
                                         const void * Transpor
                                         tClientData);

/*! \brief Initialize the UDP transport module to receive statistics.

\param TransportClientData A pointer which may be filled in (by this function) with
                           a pointer to transport-specific client data.
\param TransportOptions The TransportOptions argument originally passed to
                        lbmmon_sctl_create().
\return Zero if successful, -1 otherwise.
*/
LBMEExpDLL int lbmmon_transport_udp_initrcv(void * * TransportClientData,
                                              const void * Transpor
                                         tClientData,
                                         size_t Length,
                                         void * Transpor
                                         tClientData);

/*! \brief Send a statistics packet.

\param Data The data to be sent.
\param Length The length of the data.
\param TransportClientData A pointer to transport-specific client data.
\return Zero if successful, -1 otherwise.
*/
LBMEExpDLL int lbmmon_transport_udp_send(const char * Data,
                                          size_t Length,
                                          void * Transpor
                                         tClientData);

/*! \brief Receive statistics packet data.

\param Data A pointer to a buffer to receive the packet data.
\param Length A pointer to a size_t. On entry, it contains the maximum number of bytes
              to receive. On exit, it contains the actual number of bytes received.
\param TimeoutMS Maximum timeout in milliseconds. If no data is available within
                  the timeout value, return.
\param TransportClientData A pointer to transport-specific client data.
\return Zero if successful, -1 otherwise.
*/
LBMEExpDLL int lbmmon_transport_udp_receive(char * Data,
                                             size_t * Length,
                                             unsigned int TimeoutMS,
                                             void * Transpor
                                         tClientData);

/*! \brief Finish UDP transport module source processing.

\param TransportClientData A pointer to transport-specific client data.
\return Zero if successful, -1 otherwise.
*/
LBMEExpDLL int lbmmon_transport_udp_src_finish(void * TransportClientData);

/*! \brief Finish UDP transport module receiver processing.

\param TransportClientData A pointer to transport-specific client data.
\return Zero if successful, -1 otherwise.
*/
LBMEExpDLL int lbmmon_transport_udp_rcv_finish(void * TransportClientData);

/*! \brief Return a messages describing the last error encountered.

```

```
\return A string containing a description of the last error encountered by the module.  
*/  
LBMExpDLL const char * lbmon_transport_udp_errmsg(void);  
  
#if defined(__cplusplus)  
}  
#endif /* __cplusplus */  
  
#endif
```

## 9.7 Source code for lbmmmontrudp.c

```
/*
All of the documentation and software included in this and any
other Informatica Corporation Ultra Messaging Releases
Copyright (C) Informatica Corporation. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted only as covered by the terms of a
valid software license agreement with Informatica Corporation.

Copyright (C) 2004-2014, Informatica Corporation. All Rights Reserved.

THE SOFTWARE IS PROVIDED "AS IS" AND INFORMATICA DISCLAIMS ALL WARRANTIES
EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF
NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR
PURPOSE. INFORMATICA DOES NOT WARRANT THAT USE OF THE SOFTWARE WILL BE
UNINTERRUPTED OR ERROR-FREE. INFORMATICA SHALL NOT, UNDER ANY CIRCUMSTANCES, BE
LIABLE TO LICENSEE FOR LOST PROFITS, CONSEQUENTIAL, INCIDENTAL, SPECIAL OR
INDIRECT DAMAGES ARISING OUT OF OR RELATED TO THIS AGREEMENT OR THE
TRANSACTIONS CONTEMPLATED HEREUNDER, EVEN IF INFORMATICA HAS BEEN APPRISED OF
THE LIKELIHOOD OF SUCH DAMAGES.

*/
#ifndef __VOS__
#define _POSIX_C_SOURCE 200112L
#include <sys/time.h>
#endif

#include <stdio.h>
#include <time.h>
#include <string.h>
#include <stdlib.h>
#include <limits.h>
#include <errno.h>
#ifdef _WIN32
    #include <winsock2.h>
    #include <ws2tcpip.h>
    #define strcasecmp strcmp
    #define snprintf _snprintf
    typedef int ssize_t;
#else
    #include "config.h"
    #include <unistd.h>
    #if defined(__TANDEM)
        #if defined(HAVE_TANDEM_SPT)
            #include <ktdmtyp.h>
            #include <spthread.h>
        #else
            #include <pthread.h>
        #endif
    #else
        #include <pthread.h>
    #endif
    #include <strings.h>
    #include <sys/socket.h>

```

```
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#endif
#if defined(__VMS)
    typedef int socklen_t;
#endif
#include <lbm/lbmmmon.h>
#include <lbm/lbmmmontrudp.h>

#ifndef _WIN32
    #define LBMMON_INVALID_HANDLE INVALID_SOCKET
    #define LBMMON_SOCKET_ERROR SOCKET_ERROR
#else
    #define LBMMON_INVALID_HANDLE -1
    #define LBMMON_SOCKET_ERROR -1
#endif
#ifndef INADDR_NONE
    #define INADDR_NONE ((in_addr_t) 0xffffffff)
#endif

/*
     Package all of the needed function pointers for this module into a
     lbmmmon_transport_func_t structure.
*/
static const lbmmmon_transport_func_t LBMMON_TRANSPORT_UDP =
{
    lbmmmon_transport_udp_initsrc,
    lbmmmon_transport_udp_initrcv,
    lbmmmon_transport_udp_send,
    lbmmmon_transport_udp_receive,
    lbmmmon_transport_udp_src_finish,
    lbmmmon_transport_udp_rcv_finish,
    lbmmmon_transport_udp_errmsg
};

/*
     For a statistics source, one of these gets returned as the TransportClientData.
*/
typedef struct
{
    /* Socket used to send a statistics packet */
#ifdef _WIN32
    SOCKET mSocket;
#else
    int mSocket;
#endif
    /* Peer socket address */
    struct sockaddr_in mPeer;
    /* Mode */
    unsigned char mMode;
} lbmmmon_transport_udp_src_t;

#define MODE_UNICAST 0
#define MODE_BROADCAST 1
#define MODE_MULTICAST 2
```

```

/*
   A queue of incoming statistics packets is maintained. This describes each
   entry in the queue.
*/
struct lbmmon_transport_udp_rcv_node_t_stct
{
    /* Pointer to the message */
    unsigned char * mMessage;
    /* Length of the message */
    size_t mMessageLength;
    /* Number of bytes of the message returned to caller */
    size_t mUsedBytes;
    /* Next entry in the queue */
    struct lbmmon_transport_udp_rcv_node_t_stct * mNext;
};

typedef struct lbmmon_transport_udp_rcv_node_t_stct lbmmon_transport_udp_rcv_node_t;

/*
   For a statistics receiver, one of these gets returned as the TransportClientData.
*/
typedef struct
{
    /* Lock to prevent access by multiple threads */
#ifdef _WIN32
    CRITICAL_SECTION mLock;
#else
    pthread_mutex_t mLock;
#endif
    /* Socket used to receive packets */
#ifdef _WIN32
    SOCKET mSocket;
#else
    int mSocket;
#endif
    /* Peer socket address */
    struct sockaddr_in mPeer;
    /* Interface */
    struct sockaddr_in mInterface;
    /* Multicast membership */
    struct ip_mreq mMulticastMembership;
    /* Mode */
    unsigned char mMode;
    /* Head of the message queue */
    lbmmon_transport_udp_rcv_node_t * mHead;
    /* Tail of the message queue */
    lbmmon_transport_udp_rcv_node_t * mTail;
    /* Receiving thread */
#ifdef _WIN32
    HANDLE mThread;
#else
    pthread_t mThread;
#endif
    /* Flag to terminate thread */
    unsigned char mTerminateThread;
} lbmmon_transport_udp_rcv_t;

#define DEFAULT_INTERFACE "0.0.0.0"

```

```
#define DEFAULT_PORT "2933"
#define DEFAULT_TTL "16"

/* Error codes */
#define LBMMONTRUDP_ERR_INVALID_OPTION 1
#define LBMMONTRUDP_ERR_SOCKET 2
#define LBMMONTRUDP_ERR_SEND 3
#define LBMMONTRUDP_ERR_THREAD 4

static void lock_receiver(lbmmmon_transport_udp_rcv_t * Receiver);
static void unlock_receiver(lbmmmon_transport_udp_rcv_t * Receiver);
#ifndef _WIN32
static DWORD WINAPI receive_thread_proc(void * Arg);
#else
static void * receive_thread_proc(void * Arg);
#endif

static char ErrorString[1024];

static const char *
last_socket_error(void)
{
    static char message[512];
#ifndef _WIN32
    sprintf(message, sizeof(message), "error %d", WSAGetLastError());
#else
    sprintf(message,
            sizeof(message),
            "error %d, %s",
            errno,
            strerror(errno));
#endif
    return (message);
}

const lbmmmon_transport_func_t *
lbmmmon_transport_udp_module(void)
{
    return (&LBMMON_TRANSPORT_UDP);
}

int
lbmmmon_transport_udp_initsrc(void ** TransportClientData, const void * TransportOptions)
{
    lbmmmon_transport_udp_src_t * data;
    int rc;
    const char * ptr = (const char *) TransportOptions;
    char key[512];
    char value[512];
    char address[512];
    char port[512];
    char interface[512];
    char mcgroup[512];
    char bcaddress[512];
    char ttl[512];
    unsigned long port_value;
    struct in_addr multicast_group;
```

```

    struct in_addr multicast_interface;
    struct in_addr broadcast_address;
    struct in_addr host_address;
    unsigned long ttl_value = 0;

    memset(ErrorString, 0, sizeof(ErrorString));
    data = malloc(sizeof(lbmmon_transport_udp_src_t));
    multicast_group.s_addr = 0;
    multicast_interface.s_addr = 0;
    broadcast_address.s_addr = 0;
    host_address.s_addr = 0;

    /* Process any options */
    memset(address, 0, sizeof(address));
    memset(port, 0, sizeof(port));
    strcpy(port, DEFAULT_PORT);
    memset(interface, 0, sizeof(interface));
    strcpy(interface, DEFAULT_INTERFACE);
    memset(mcgroup, 0, sizeof(mcgroup));
    memset(bcaddress, 0, sizeof(bcaddress));
    memset(ttl, 0, sizeof(ttl));
    strcpy(ttl, DEFAULT_TTL);
    data->mMode = MODE_UNICAST;
    while ((ptr = lbmmon_next_key_value_pair(ptr, key, sizeof(key), value, sizeof(value))) != NULL)
    {
        if (strcasecmp(key, "address") == 0)
        {
            strncpy(address, value, sizeof(address));
        }
        else if (strcasecmp(key, "port") == 0)
        {
            strncpy(port, value, sizeof(port));
        }
        else if (strcasecmp(key, "interface") == 0)
        {
            strncpy(interface, value, sizeof(interface));
        }
        else if (strcasecmp(key, "mcgroup") == 0)
        {
            strncpy(mcgroup, value, sizeof(mcgroup));
        }
        else if (strcasecmp(key, "bcaddress") == 0)
        {
            strncpy(bcaddress, value, sizeof(bcaddress));
        }
        else if (strcasecmp(key, "ttl") == 0)
        {
            strncpy(ttl, value, sizeof(ttl));
        }
    }

    /* Validate the options
     * Note the following:
     * - interface and ttl only apply to mcgroup
     * - mcgroup (and thus multicast) takes precedence over bcaddress (and thus broadcast)
     * - bcaddress takes precedence over address.
     */

```

```
port_value = strtoul(port, NULL, 0);
if ((port_value == ULONG_MAX) && (errno == ERANGE))
{
    strncpy(ErrorString, "Invalid port value", sizeof(ErrorString));
    free(data);
    return (LBMMONTRUDP_ERR_INVALID_OPTION);
}
else if (port_value > USHRT_MAX)
{
    strncpy(ErrorString, "Invalid port value", sizeof(ErrorString));
    free(data);
    return (LBMMONTRUDP_ERR_INVALID_OPTION);
}

if (mcgroup[0] != '\0')
{
    data->mMode = MODE_MULTICAST;
    multicast_group.s_addr = inet_addr(mcgroup);
    if (multicast_group.s_addr == INADDR_NONE)
    {
        strncpy(ErrorString, "Invalid mcgroup value", sizeof(ErrorString));
        free(data);
        return (LBMMONTRUDP_ERR_INVALID_OPTION);
    }
    if (!IN_MULTICAST(ntohl(multicast_group.s_addr)))
    {
        strncpy(ErrorString, "Invalid mcgroup value", sizeof(ErrorString));
        free(data);
        return (LBMMONTRUDP_ERR_INVALID_OPTION);
    }
    multicast_interface.s_addr = inet_addr(interface);
    if (multicast_interface.s_addr == INADDR_NONE)
    {
        strncpy(ErrorString, "Invalid interface value", sizeof(ErrorString));
        free(data);
        return (LBMMONTRUDP_ERR_INVALID_OPTION);
    }
    ttl_value = strtoul(ttl, NULL, 0);
    if ((ttl_value == ULONG_MAX) && (errno == ERANGE))
    {
        strncpy(ErrorString, "Invalid ttl value", sizeof(ErrorString));
        free(data);
        return (LBMMONTRUDP_ERR_INVALID_OPTION);
    }
    else if (ttl_value > UCHAR_MAX)
    {
        strncpy(ErrorString, "Invalid ttl value", sizeof(ErrorString));
        free(data);
        return (LBMMONTRUDP_ERR_INVALID_OPTION);
    }
}
else if (bcaddress[0] != '\0')
{
    data->mMode = MODE_BROADCAST;
    broadcast_address.s_addr = inet_addr(bcaddress);
    if (broadcast_address.s_addr == INADDR_NONE)
    {
```

```

        strncpy(ErrorString, "Invalid bcaddress value", sizeof(ErrorString));
        free(data);
        return (LBMMONTRUDP_ERR_INVALID_OPTION);
    }
}
else
{
    host_address.s_addr = inet_addr(address);
    if (host_address.s_addr == INADDR_NONE)
    {
        strncpy(ErrorString, "Invalid address value", sizeof(ErrorString));
        free(data);
        return (LBMMONTRUDP_ERR_INVALID_OPTION);
    }
}

/* Create the socket */
data->mSocket = socket(PF_INET, SOCK_DGRAM, 0);
if (data->mSocket == LBMMON_INVALID_HANDLE)
{
    snprintf(ErrorString,
              sizeof(ErrorString),
              "socket() failed, %s",
              last_socket_error());
    free(data);
    return (LBMMONTRUDP_ERR_SOCKET);
}

/* If broadcast mode, enable broadcast on the socket */
if (data->mMode == MODE_BROADCAST)
{
    int option = 1;
    socklen_t len = sizeof(option);
    rc = setsockopt(data->mSocket, SOL_SOCKET, SO_BROADCAST, (void *) &option, len);
    if (rc == LBMMON_SOCKET_ERROR)
    {
        snprintf(ErrorString,
                  sizeof(ErrorString),
                  "setsockopt(...,SO_BROADCAST,...) failed, %s",
                  last_socket_error());
    }
}

#endif _WIN32
else
#endif
{
    free(data);
    return (LBMMONTRUDP_ERR_SOCKET);
}

/* For multicast, set the outgoing interface and TTL. */
if (data->mMode == MODE_MULTICAST)
{
    unsigned char optval = (unsigned char) ttl_value;
    struct in_addr ifc_addr;
    rc = setsockopt(data->mSocket, IPPROTO_IP, IP_MULTICAST_TTL, (void *) &optval,
                    if (rc != LBMMON_SOCKET_ERROR)

```

```

{
    ifc_addr.s_addr = multicast_interface.s_addr;
    rc = setsockopt(data->mSocket, IPPROTO_IP, IP_MULTICAST_IF, (void *) &ifc_addr, si
if (rc == LBMMON_SOCKET_ERROR)
{
    snprintf(ErrorString,
             sizeof(ErrorString),
             "setsockopt(...,IP_MULTICAST_IF,...) failed, %s",
             last_socket_error());
}
else
{
    snprintf(ErrorString,
             sizeof(ErrorString),
             "setsockopt(...,IP_MULTICAST_TTL,...) failed, %s",
             last_socket_error());
}
if (rc == LBMMON_SOCKET_ERROR)
{
#endif _WIN32
    closesocket(data->mSocket);
#else
    close(data->mSocket);
#endif
    free(data);
    return (LBMMONTRUDP_ERR_SOCKET);
}
}

/* Build the peer sockaddr_in. */
data->mPeer.sin_family = AF_INET;
data->mPeer.sin_port = htons((unsigned short) port_value);
switch (data->mMode)
{
    case MODE_UNICAST:
    default:
        data->mPeer.sin_addr.s_addr = host_address.s_addr;
        break;

    case MODE_BROADCAST:
        data->mPeer.sin_addr.s_addr = broadcast_address.s_addr;
        break;

    case MODE_MULTICAST:
        data->mPeer.sin_addr.s_addr = multicast_group.s_addr;
        break;
}

/* Pass back the lbmmmon_transport_udp_src_t created */
*TransportClientData = data;
return (0);
}

int
lbmmmon_transport_udp_initrcv(void * * TransportClientData, const void * TransportOptions)
{

```

```

lbmmmon_transport_udp_rcv_t * data;
int rc;
const char * ptr = (const char *) TransportOptions;
char key[512];
char value[512];
char port[512];
char interface[512];
char mcgroup[512];
unsigned long port_value;
struct in_addr multicast_group;
struct in_addr multicast_interface;

memset(ErrorString, 0, sizeof(ErrorString));
data = malloc(sizeof(lbmmmon_transport_udp_rcv_t));
multicast_group.s_addr = 0;
multicast_interface.s_addr = 0;
data->mHead = NULL;
data->mTail = NULL;
data->mTerminateThread = 0;

/* Process any options */
memset(port, 0, sizeof(port));
strcpy(port, DEFAULT_PORT);
memset(interface, 0, sizeof(interface));
strcpy(interface, DEFAULT_INTERFACE);
memset(mcgroup, 0, sizeof(mcgroup));
data->mMode = MODE_UNICAST;
while ((ptr = lbmmmon_next_key_value_pair(ptr, key, sizeof(key), value, sizeof(value))) != NULL)
{
    if (strcasecmp(key, "port") == 0)
    {
        strncpy(port, value, sizeof(port));
    }
    else if (strcasecmp(key, "interface") == 0)
    {
        strncpy(interface, value, sizeof(interface));
    }
    else if (strcasecmp(key, "mcgroup") == 0)
    {
        strncpy(mcgroup, value, sizeof(mcgroup));
    }
}

/* Validate the options
Note the following:
- interface only applies to mcgroup
- mcgroup (and thus multicast) takes precedence over broadcast/unicast
*/
port_value = strtoul(port, NULL, 0);
if ((port_value == ULONG_MAX) && (errno == ERANGE))
{
    strncpy(ErrorString, "Invalid port value", sizeof(ErrorString));
    free(data);
    return (LBMMONTRUDP_ERR_INVALID_OPTION);
}
else if (port_value > USHRT_MAX)
{

```

```

        strncpy(ErrorString, "Invalid port value", sizeof(ErrorString));
        free(data);
        return (LBMMONTRUDP_ERR_INVALID_OPTION);
    }

    if (mcgroup[0] != '\0')
    {
        data->mMode = MODE_MULTICAST;
        multicast_group.s_addr = inet_addr(mcgroup);
        if (multicast_group.s_addr == INADDR_NONE)
        {
            strncpy(ErrorString, "Invalid mcgroup value", sizeof(ErrorString));
            free(data);
            return (LBMMONTRUDP_ERR_INVALID_OPTION);
        }
        if (!IN_MULTICAST(ntohl(multicast_group.s_addr)))
        {
            strncpy(ErrorString, "Invalid mcgroup value", sizeof(ErrorString));
            free(data);
            return (LBMMONTRUDP_ERR_INVALID_OPTION);
        }
        multicast_interface.s_addr = inet_addr(interface);
        if (multicast_interface.s_addr == INADDR_NONE)
        {
            strncpy(ErrorString, "Invalid interface value", sizeof(ErrorString));
            free(data);
            return (LBMMONTRUDP_ERR_INVALID_OPTION);
        }
    }

/* Create the socket */
data->mSocket = socket(PF_INET, SOCK_DGRAM, 0);
if (data->mSocket == LBMMON_INVALID_HANDLE)
{
    snprintf(ErrorString,
             sizeof(ErrorString),
             "socket() failed, %s",
             last_socket_error());
    free(data);
    return (LBMMONTRUDP_ERR_SOCKET);
}

/* Build the interface sockaddr_in. */
memset(&(data->mInterface), 0, sizeof(data->mInterface));
data->mInterface.sin_family = AF_INET;
data->mInterface.sin_port = htons((unsigned short) port_value);
data->mInterface.sin_addr.s_addr = INADDR_ANY;

/* Bind the socket. */
rc = bind(data->mSocket, (struct sockaddr *) &(data->mInterface), sizeof(data->mInterface));
if (rc == LBMMON_SOCKET_ERROR)
{
    snprintf(ErrorString,
             sizeof(ErrorString),
             "bind() failed, %s",
             last_socket_error());
}

#endif _WIN32

```

```

        closesocket(data->mSocket);
#else
        close(data->mSocket);
#endif
        return (LBMMONTRUDP_ERR_SOCKET);
    }

/* For multicast, join the group. */
if (data->mMode == MODE_MULTICAST)
{
    data->mMulticastMembership.imr_interface.s_addr = multicast_interface.s_addr;
    data->mMulticastMembership.imr_multiaddr.s_addr = multicast_group.s_addr;
    rc = setsockopt(data->mSocket, IPPROTO_IP, IP_ADD_MEMBERSHIP, (void *) &(data-
    if (rc == LBMON_SOCKET_ERROR)
    {
        sprintf(ErrorString,
                sizeof(ErrorString),
                "setsockopt(...,IP_ADD_MEMBERSHIP,...) failed, %s",
                last_socket_error());
    }
    #ifdef _WIN32
        closesocket(data->mSocket);
    #else
        close(data->mSocket);
    #endif
    free(data);
    return (LBMMONTRUDP_ERR_SOCKET);
}
}

/* Build the peer sockaddr_in. */
data->mPeer.sin_family = AF_INET;
data->mPeer.sin_port = htons((unsigned short) port_value);
data->mPeer.sin_addr.s_addr = INADDR_ANY;

#endif _WIN32
    InitializeCriticalSection(&(data->mLock));
#else
    pthread_mutex_init(&(data->mLock), NULL);
#endif

/* Start the receive thread */
#endif _WIN32
    data->mThread = CreateThread(NULL, 0, receive_thread_proc, data, 0, NULL);
    if (data->mThread == NULL)
    {
        sprintf(ErrorString,
                sizeof(ErrorString),
                "CreateThread() failed, error %d",
                GetLastError());
        closesocket(data->mSocket);
        free(data);
        return (LBMMONTRUDP_ERR_THREAD);
    }
}
#endif
#endif __VOS__
{
    pthread_attr_t pth_attr;

```

```

        pthread_attr_init (&pth_attr);
        pthread_attr_setschedpolicy (&pth_attr, SCHED_RR);
        rc = pthread_create(&(data->mThread), &pth_attr, receive_thread_proc, data);
    }
#endif
    else
        rc = pthread_create(&(data->mThread), NULL, receive_thread_proc, data);
#endif
    if (rc != 0)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),
                 "pthread_create() failed, error %d, %s",
                 rc,
                 strerror(rc));
        close(data->mSocket);
        free(data);
        return (LBMMONTRUDP_ERR_THREAD);
    }
#endif

/* Pass back the lbmmmon_transport_udp_rcv_t created */
*TransportClientData = data;
return (0);
}

#ifndef _WIN32
DWORD WINAPI
receive_thread_proc(void * Arg)
#else
void *
receive_thread_proc(void * Arg)
#endif
{
    lbmmmon_transport_udp_rcv_t * rcv = (lbmmmon_transport_udp_rcv_t *) Arg;
    unsigned char buffer[8192];
    struct timeval timeout;
    fd_set readfds;
    int rc;
    ssize_t bytes_read;
    lbmmmon_transport_udp_rcv_node_t * node;

    while (rcv->mTerminateThread == 0)
    {
        FD_ZERO(&readfds);
        FD_SET(rcv->mSocket, &readfds);
        timeout.tv_sec = 0;
        timeout.tv_usec = 500000;
        rc = select(rcv->mSocket + 1, &readfds, NULL, NULL, &timeout);
        if (rc <= 0)
        {
            continue;
        }
        bytes_read = recvfrom(rcv->mSocket, buffer, sizeof(buffer), 0, NULL, NULL);
        if (bytes_read == LBMMON_SOCKET_ERROR)
        {
            continue;
        }
    }
}

```

```

/* A data message. We want to enqueue it for processing. */
lock_receiver(rcv);
node = malloc(sizeof(lbmmon_transport_udp_rcv_node_t));
node->mMessage = malloc((size_t) bytes_read);
memcpy(node->mMessage, buffer, (size_t) bytes_read);
node->mMessageLength = (size_t) bytes_read;
node->mUsedBytes = 0; /* No data returned as yet */

/* Link the message onto the queue */
node->mNext = NULL;
if (rcv->mTail != NULL)
{
    rcv->mTail->mNext = node;
}
else
{
    rcv->mHead = node;
}
rcv->mTail = node;
unlock_receiver(rcv);

#endif /* _WIN32 */
return (0);
#else
return (NULL);
#endif
}

int
lbmmmon_transport_udp_send(const char * Data, size_t Length, void * TransportClientData)
{
    lbmmmon_transport_udp_src_t * src;
    int rc;

    if ((Data == NULL) || (TransportClientData == NULL))
    {
        return (-1);
    }
    src = (lbmmmon_transport_udp_src_t *) TransportClientData;
    rc = sendto(src->mSocket, Data, Length, 0, (struct sockaddr *) &(src->mPeer), sizeof(struct sockaddr));
    if (rc == LBMMON_SOCKET_ERROR)
    {
        sprintf(ErrorString,
                sizeof(ErrorString),
                "sendto() failed, %s",
                last_socket_error());
        return (LBMMONTRUDP_ERR_SEND);
    }
    return (0);
}

int
lbmmmon_transport_udp_receive(char * Data, size_t * Length, unsigned int TimeoutMS, void * TransportClientData)
{
    lbmmmon_transport_udp_rcv_t * rcv = (lbmmmon_transport_udp_rcv_t *) TransportClientData;
    lbmmmon_transport_udp_rcv_node_t * node;
}

```

```

        int rc = 0;
        size_t length_remaining;
#if defined(_WIN32)
#elif defined(__TANDEM)
        unsigned int sleep_sec;
        unsigned int sleep_usec;
#else
        struct timespec ivl;
#endif

        if ((Data == NULL) || (Length == NULL) || (TransportClientData == NULL))
        {
            return (-1);
        }
        if (*Length == 0)
        {
            return (0);
        }
        lock_receiver(rcv);
        if (rcv->mHead != NULL)
        {
            /* Queue is non-empty. Pull the first message from the queue. */
            node = rcv->mHead;
            length_remaining = node->mMessageLength - node->mUsedBytes;
            if (*Length >= length_remaining)
            {
                /* We can transfer the rest of the message */
                memcpy(Data, node->mMessage + node->mUsedBytes, length_remaining);
                *Length = length_remaining;
                rc = 0;
                /* We're done with the message, so free it. */
                free(node->mMessage);
                node->mMessage = NULL;
                /* Unlink the node from the queue */
                rcv->mHead = node->mNext;
                if (rcv->mHead == NULL)
                {
                    rcv->mTail = NULL;
                }
                free(node);
            }
            else
            {
                /* Can only transfer part of the message */
                memcpy(Data, node->mMessage + node->mUsedBytes, *Length);
                node->mUsedBytes -= *Length;
                rc = 0;
            }
            unlock_receiver(rcv);
        }
        else
        {
            unlock_receiver(rcv);
            /* Sleep for wait time */
#define NANOSECONDS_PER_SECOND 10000000000
#define MICROSECONDS_PER_SECOND 1000000
#define MILLISECONDS_PER_SECOND 1000

```

```

#define NANOSECONDS_PER_MILLISECOND (NANOSECONDS_PER_SECOND / MILLISECONDS_PER_SECOND)
#define MICROSECONDS_PER_MILLISECOND (MICROSECONDS_PER_SECOND / MILLISECONDS_PER_SECOND)
#if defined(_WIN32)
    Sleep(TimeoutMS);
#elif defined(__TANDEM)
    sleep_sec = TimeoutMS / MILLISECONDS_PER_SECOND;
    sleep_usec = (TimeoutMS % MILLISECONDS_PER_SECOND) * MICROSECONDS_PER_MILLISECOND;
    if (sleep_usec > 0)
    {
        usleep(sleep_usec);
    }
    if (sleep_sec > 0)
    {
        sleep(sleep_sec);
    }
#else
    ivl.tv_sec = TimeoutMS / MILLISECONDS_PER_SECOND;
    ivl.tv_nsec = (TimeoutMS % MILLISECONDS_PER_SECOND) * NANOSECONDS_PER_MILLISECOND;
    nanosleep(&ivl, NULL);
#endif
    rc = 1;
}
return (rc);
}

int
lbmmmon_transport_udp_src_finish(void * TransportClientData)
{
    lbmmmon_transport_udp_src_t * src;

    if (TransportClientData == NULL)
    {
        strncpy(ErrorString, "Invalid parameter", sizeof(ErrorString));
        return (-1);
    }
    src = (lbmmmon_transport_udp_src_t *) TransportClientData;

#ifdef _WIN32
    closesocket(src->mSocket);
#else
    close(src->mSocket);
#endif
    /* Clean up our data */
    free(TransportClientData);
    return (0);
}

int
lbmmmon_transport_udp_rcv_finish(void * TransportClientData)
{
    lbmmmon_transport_udp_rcv_t * rcv = (lbmmmon_transport_udp_rcv_t *) TransportClientData;
    lbmmmon_transport_udp_rcv_node_t * node;
    lbmmmon_transport_udp_rcv_node_t * next;
    int rc;

    /* Stop the thread to prevent any more incoming messages */
    rcv->mTerminateThread = 1;
}

```

```

/* Lock the receiver */
lock_receiver(rcv);

/* Clean out the queue */
node = rcv->mHead;
while (node != NULL)
{
    /* Let LBM know we're done with the message */
    free(node->mMessage);
    next = node->mNext;
    free(node);
    node = next;
}

unlock_receiver(rcv);

/* If multicast, drop membership. */
if (rcv->mMode == MODE_MULTICAST)
{
    rc = setsockopt(rcv->mSocket, IPPROTO_IP, IP_DROP_MEMBERSHIP, (void *) &(rcv->mMulticastMe
}
#endif _WIN32
closesocket(rcv->mSocket);
#else
close(rcv->mSocket);
#endif
#endif _WIN32
DeleteCriticalSection(&(rcv->mLock));
#else
pthread_mutex_destroy(&(rcv->mLock));
#endif

free(TransportClientData);
return (0);
}

void
lock_receiver(lbmmmon_transport_udp_rcv_t * Receiver)
{
#endif _WIN32
    EnterCriticalSection(&(Receiver->mLock));
#else
    pthread_mutex_lock(&(Receiver->mLock));
#endif
}

void
unlock_receiver(lbmmmon_transport_udp_rcv_t * Receiver)
{
#endif _WIN32
    LeaveCriticalSection(&(Receiver->mLock));
#else
    pthread_mutex_unlock(&(Receiver->mLock));
#endif
}

```

```
const char *
lbmmmon_transport_udp_errmsg(void)
{
    return (ErrorString);
}
```

## 9.8 LBMMON CSV format module

- [lbmmonfmtcsv.h](#)
- [lbmmonfmtcsv.c](#)

## 9.9 Source code for lbmmmonfmtcsv.h

```
/** \file lbmmmonfmtcsv.h
 \brief Ultra Messaging (UM) Monitoring API
 \author David K. Ameiss - Informatica Corporation
 \version $Id: //UMprod/REL_5_3_6/29West/lbm/src/mon/lbm/lbmmmonfmtcsv.h#2 $

The Ultra Messaging (UM) Monitoring API Description. Included
are types, constants, and functions related to the API. Contents are
subject to change.

All of the documentation and software included in this and any
other Informatica Corporation Ultra Messaging Releases
Copyright (C) Informatica Corporation. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted only as covered by the terms of a
valid software license agreement with Informatica Corporation.

Copyright (C) 2006-2014, Informatica Corporation. All Rights Reserved.

THE SOFTWARE IS PROVIDED "AS IS" AND INFORMATICA DISCLAIMS ALL WARRANTIES
EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF
NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR
PURPOSE. INFORMATICA DOES NOT WARRANT THAT USE OF THE SOFTWARE WILL BE
UNINTERRUPTED OR ERROR-FREE. INFORMATICA SHALL NOT, UNDER ANY CIRCUMSTANCES, BE
LIABLE TO LICENSEE FOR LOST PROFITS, CONSEQUENTIAL, INCIDENTAL, SPECIAL OR
INDIRECT DAMAGES ARISING OUT OF OR RELATED TO THIS AGREEMENT OR THE
TRANSACTIONS CONTEMPLATED HEREUNDER, EVEN IF INFORMATICA HAS BEEN APPRISED OF
THE LIKELIHOOD OF SUCH DAMAGES.

*/
#ifndef LBMMONFMTCSV_H
#define LBMMONFMTCSV_H

#include <stdlib.h>
#ifdef _WIN32
    #include <winsock2.h>
#endif
#include <lbm/lbmmmon.h>

#if defined(__cplusplus)
extern "C" {
#endif /* __cplusplus */

/*! \brief Return a pointer to the LBMMON_FORMAT_CSV module structure.

\return Pointer to LBMMON_FORMAT_CSV.
*/
LBMDLL const lbmmmon_format_func_t * lbmmmon_format_csv_module(void);

/*! \brief Initialize the CSV format module.

\param FormatClientData A pointer which may be filled in
(by this function)
with a pointer to format-specific client data.
\param FormatOptions The FormatOptions argument originally passed to

```

```

        lbmmmon_sctl_create() or lbmmmon_rctl_create().
\return Zero if successful, -1 otherwise.
If -1 is returned,
the serialized data will not be sent.
*/
LBMEExpDLL int lbmmmon_format_csv_init(void * * FormatClientData,
                                         const void * FormatOptions);

/*! \brief Serialize an ::lbt_rcv_transport_stats_t structure to CSV.

\param Destination A pointer to a buffer to receive the serialized format
of the ::lbt_rcv_transport_stats_t statistics.
\param Size A pointer to a \c size_t.
On entry,
it contains the maximum allowed size of the serialized statistics.
On exit,
it must contain the actual size of the serialized statistics.
\param ModuleID A pointer to an \c unsigned \c short,
into which the module may write a module identification value.
This value is included in the transmitted packet,
and may be used by the receiver to verify and differentiate between
different version of the module (and thus the format of the data it expects).
\param Statistics A pointer to an ::lbt_rcv_transport_stats_t structure
to be serialized.
\param FormatClientData A pointer to format-specific client data.
\return Zero if successful, -1 otherwise.
If -1 is returned,
the serialized data will not be sent.
*/
LBMEExpDLL int lbmmmon_rcv_format_csv_serialize(char * Destination,
                                                 size_t * Size,
                                                 unsigned short * ModuleID,
                                                 const lbt_rcv_transport_stats_t * Statistics,
                                                 void * FormatClientData);

/*! \brief Serialize an ::lbt_src_transport_stats_t structure to CSV.

\param Destination A pointer to a buffer to receive the serialized format
of the ::lbt_src_transport_stats_t statistics.
\param Size A pointer to a \c size_t.
On entry,
it contains the maximum allowed size of the serialized statistics.
On exit,
it must contain the actual size of the serialized statistics.
\param ModuleID A pointer to an \c unsigned \c short,
into which the module may write a module identification value.
This value is included in the transmitted packet,
and may be used by the receiver to verify and differentiate between
different version of the module (and thus the format of the data it expects).
\param Statistics A pointer to an ::lbt_src_transport_stats_t structure to be
serialized.
\param FormatClientData A pointer to format-specific client data.
\return Zero if successful, -1 otherwise.
If -1 is returned,
the serialized data will not be sent.
*/
LBMEExpDLL int lbmmmon_src_format_csv_serialize(char * Destination,
                                                 size_t * Size,
                                                 unsigned short * ModuleID,
                                                 const lbt_src_transport_stats_t * Statistics,
                                                 void * FormatClientData);

```

```

size_
unsig
const
void

/*! \brief Serialize an ::lbt_event_queue_stats_t structure to CSV.

\param Destination A pointer to a buffer to receive the serialized format
of the ::lbt_event_queue_stats_t statistics.
\param Size A pointer to a \c size_t.
    On entry,
    it contains the maximum allowed size of the serialized statistics.
    On exit,
    it must contain the actual size of the serialized statistics.
\param ModuleID A pointer to an \c unsigned \c short,
    into which the module may write a module identification value.
    This value is included in the transmitted packet,
    and may be used by the receiver to verify and differentiate between
    different version of the module (and thus the format of the data it expects).
\param Statistics A pointer to an ::lbt_event_queue_stats_t structure
    to be serialized.
\param FormatClientData A pointer to format-specific client data.
\return Zero if successful, -1 otherwise.
    If -1 is returned,
    the serialized data will not be sent.
*/
LBMEExpDLL int lbmmon_evt_format_csv_serialize(char * Destination,
size_
unsig
const
void

/*! \brief Serialize an ::lbt_context_stats_t structure to CSV.

\param Destination A pointer to a buffer to receive the serialized format
of the ::lbt_context_stats_t statistics.
\param Size A pointer to a \c size_t.
    On entry,
    it contains the maximum allowed size of the serialized statistics.
    On exit,
    it must contain the actual size of the serialized statistics.
\param ModuleID A pointer to an \c unsigned \c short,
    into which the module may write a module identification value.
    This value is included in the transmitted packet,
    and may be used by the receiver to verify and differentiate between
    different version of the module (and thus the format of the data it expects).
\param Statistics A pointer to an ::lbt_context_stats_t structure
    to be serialized.
\param FormatClientData A pointer to format-specific client data.
\return Zero if successful, -1 otherwise.
    If -1 is returned,
    the serialized data will not be sent.
*/
LBMEExpDLL int lbmmon_ctx_format_csv_serialize(char * Destination,
size_
unsig
const

```

```

void * FormatCli

/*!
 \brief Deserialize a buffer from CSV into an ::lbm_rcv_transport_stats_t
        structure.

 \param Statistics A pointer to an ::lbm_rcv_transport_stats_t structure into
                  which the data is serialized.
 \param Source A pointer to a buffer containing the serialized data.
 \param Length The length of the serialized data.
 \param ModuleID The module ID received in the packet.
                  It may be used to verify and differentiate between different version of
                  the module (and thus the format of the data it expects).
 \param FormatClientData A pointer to format-specific client data.
 \return Zero if successful, -1 otherwise.
         If -1 is returned,
                 the deserialized data will not be delivered to the application.
*/
LBMDLL int lbmon_rcv_format_csv_deserialize(lbm_rcv_transport_stats_t * Statistics,
                                              const char * Source,
                                              size_t Length,
                                              unsigned short ModuleID,
                                              void * FormatClientData);

/*!
 \brief Deserialize a buffer from CSV into an ::lbm_src_transport_stats_t
        structure.

 \param Statistics A pointer to an ::lbm_src_transport_stats_t structure into
                  which the data is serialized.
 \param Source A pointer to a buffer containing the serialized data.
 \param Length The length of the serialized data.
 \param ModuleID The module ID received in the packet.
                  It may be used to verify and differentiate between different version of
                  the module (and thus the format of the data it expects).
 \param FormatClientData A pointer to format-specific client data.
 \return Zero if successful, -1 otherwise.
         If -1 is returned,
                 the deserialized data will not be delivered to the application.
*/
LBMDLL int lbmon_src_format_csv_deserialize(lbm_src_transport_stats_t * Statistics,
                                              const char * Source,
                                              size_t Length,
                                              unsigned short ModuleID,
                                              void * FormatClientData);

/*!
 \brief Deserialize a buffer from CSV into an ::lbm_event_queue_stats_t
        structure.

 \param Statistics A pointer to an ::lbm_event_queue_stats_t structure into
                  which the data is serialized.
 \param Source A pointer to a buffer containing the serialized data.
 \param Length The length of the serialized data.
 \param ModuleID The module ID received in the packet.
                  It may be used to verify and differentiate between different version of
                  the module (and thus the format of the data it expects).
 \param FormatClientData A pointer to format-specific client data.
 \return Zero if successful, -1 otherwise.
         If -1 is returned,
                 the deserialized data will not be delivered to the application.
*/
LBMDLL int lbmon_event_format_csv_deserialize(lbm_event_queue_stats_t * Statistics,
                                                const char * Source,
                                                size_t Length,
                                                unsigned short ModuleID,
                                                void * FormatClientData);

```

```

        the deserialized data will not be delivered to the application.
*/
LBMErpDLL int lbmmmon_evq_format_csv_deserialize(lbm_event_queue_stats_t * Statistics,
                                                   void * FormatClientData,
                                                   int Length,
                                                   int ModuleID);

        \brief Deserialize a buffer from CSV into an ::lbm_context_stats_t
        structure.

        \param Statistics A pointer to an ::lbm_context_stats_t structure into
                          which the data is deserialized.
        \param Source A pointer to a buffer containing the serialized data.
        \param Length The length of the serialized data.
        \param ModuleID The module ID received in the packet.
                          It may be used to verify and differentiate between different version of
                          the module (and thus the format of the data it expects).
        \param FormatClientData A pointer to format-specific client data.
        \return Zero if successful, -1 otherwise.
                If -1 is returned,
                the deserialized data will not be delivered to the application.
*/
LBMErpDLL int lbmmmon_ctx_format_csv_deserialize(lbm_context_stats_t * Statistics,
                                                   void * FormatClientData,
                                                   int Length,
                                                   int ModuleID);

        \brief Finish CSV format module processing.

        \param FormatClientData A pointer to format-specific client data.
        \return Zero if successful, -1 otherwise.
                If -1 is returned,
                the serialized data will not be sent.
*/
LBMErpDLL int lbmmmon_format_csv_finish(void * FormatClientData);

        \brief Return a messages describing the last error encountered.

        \return A string containing a description of the last error encountered by the module.
*/
LBMErpDLL const char * lbmmmon_format_csv_errmsg(void);

#if defined(__cplusplus)
}
#endif /* __cplusplus */

#endif

```

## 9.10 Source code for lbmmmonfmtcsv.c

```
/*
All of the documentation and software included in this and any
other Informatica Corporation Ultra Messaging Releases
Copyright (C) Informatica Corporation. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted only as covered by the terms of a
valid software license agreement with Informatica Corporation.

Copyright (C) 2004-2014, Informatica Corporation. All Rights Reserved.

THE SOFTWARE IS PROVIDED "AS IS" AND INFORMATICA DISCLAIMS ALL WARRANTIES
EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF
NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR
PURPOSE. INFORMATICA DOES NOT WARRANT THAT USE OF THE SOFTWARE WILL BE
UNINTERRUPTED OR ERROR-FREE. INFORMATICA SHALL NOT, UNDER ANY CIRCUMSTANCES, BE
LIABLE TO LICENSEE FOR LOST PROFITS, CONSEQUENTIAL, INCIDENTAL, SPECIAL OR
INDIRECT DAMAGES ARISING OUT OF OR RELATED TO THIS AGREEMENT OR THE
TRANSACTIONS CONTEMPLATED HEREUNDER, EVEN IF INFORMATICA HAS BEEN APPRISED OF
THE LIKELIHOOD OF SUCH DAMAGES.

*/
#ifndef __VOS__
#define _POSIX_C_SOURCE 200112L
#include <sys/time.h>
#endif

#include <stdio.h>
#include <string.h>
#include <stddef.h>
#include <time.h>
#include <limits.h>
#include <errno.h>
#ifndef _WIN32
    #define strcasecmp strcmp
    #define snprintf _snprintf
#else
#endif
#if defined(__TANDEM)
    #include <strings.h>
#endif
#include <lbm/lbmmmon.h>
#include <lbm/lbmmmonfmtcsv.h>
static const lbmmmon_format_func_t LBMMON_FORMAT_CSV =
{
    lbmmmon_format_csv_init,
    lbmmmon_rcv_format_csv_serialize,
    lbmmmon_src_format_csv_serialize,
    lbmmmon_rcv_format_csv_deserialize,
    lbmmmon_src_format_csv_deserialize,
    lbmmmon_format_csv_finish,
    lbmmmon_format_csv_errmsg,
    lbmmmon_evq_format_csv_serialize,
    lbmmmon_evq_format_csv_deserialize,
```

```

        lbmmmon_ctx_format_csv_serialize,
        lbmmmon_ctx_format_csv_deserialize
    };

typedef struct
{
    unsigned char mSeparator;
    size_t mBufferSize;
    char * mBuffer;
} lbmmmon_format_csv_t;

static const char * next_csv_value(const char * String, char * Value, size_t Size, char Separat

#define LBMMON_FORMAT_CSV_MODULE_ID      1
#define LBMMON_FORMAT_CSV_VERSION_1 1
#define LBMMON_FORMAT_CSV_VERSION_2 2
#define LBMMON_FORMAT_CSV_VERSION_3 3
#define LBMMON_FORMAT_CSV_VERSION_4 4
#define LBMMON_FORMAT_CSV_VERSION_CURRENT LBMMON_FORMAT_CSV_VERSION_4
#define MAKE_MODULE_VERSION(version) ((unsigned short) (((unsigned char) LBMMON_FORMAT_CSV_MODULE_ID) \
#define MODULE_ID(id) ((unsigned char) ((id & 0xff00) >> 8))
#define MODULE_VERSION(id) ((unsigned char) (id & 0xff))

typedef struct
{
    const size_t * layout;
    size_t count;
} lbmmmon_csv_layout_t;

static char ErrorString[1024];

const lbmmmon_format_func_t *
lbmmmon_format_csv_module(void)
{
    return (&LBMMON_FORMAT_CSV);
}

int
lbmmmon_format_csv_init(void ** FormatClientData,
                        const void * FormatOptions)
{
    char key[512];
    char value[512];
    const char * ptr = (const char *) FormatOptions;
    lbmmmon_format_csv_t * data;

    memset(ErrorString, 0, sizeof(ErrorString));
    data = malloc(sizeof(lbmmmon_format_csv_t));
    data->mSeparator = ',';
    data->mBufferSize = 1024;
    data->mBuffer = malloc(data->mBufferSize);

    while ((ptr = lbmmmon_next_key_value_pair(ptr, key, sizeof(key), value, sizeof(value))) != NULL)
    {
        if (strcasecmp(key, "separator") == 0)
        {
            data->mSeparator = value[0];
        }
    }
}

```

```
        }
    }
    *FormatClientData = (void *)data;
    return (0);
}

/*
Format of the CSV receiver statistics data is:
  type (one of LBM_TRANSPORT_STAT_*)
  source (as a string)
  actual statistics, depending on type....
  for LBTRM:
    msgs_rcved
    bytes_rcved
    nak_pckts_sent
    naks_sent
    lost
    ncfs_ignored
    ncfs_shed
    ncfs_rx_delay
    ncfs_unknown
    nak_stm_min
    nak_stm_mean
    nak_stm_max
    nak_tx_min
    nak_tx_mean
    nak_tx_max
    duplicate_data
    unrecovered_txw
    unrecovered_tmo
    lbm_msgs_rcved
    lbm_msgs_no_topic_rcved
    lbm_reqs_rcved
      dgrams_dropped_size
      dgrams_dropped_type
      dgrams_dropped_version
      dgrams_dropped_hdr
      dgrams_dropped_other
    out_of_order
  for LBTRU:
    msgs_rcved
    bytes_rcved
    nak_pckts_sent
    naks_sent
    lost
    ncfs_ignored
    ncfs_shed
    ncfs_rx_delay
    ncfs_unknown
    nak_stm_min
    nak_stm_mean
    nak_stm_max
    nak_tx_min
    nak_tx_mean
    nak_tx_max
    duplicate_data
    unrecovered_txw
```

```

unrecovered_tmo
lbm_msgs_rcved
lbm_msgs_no_topic_rcved
lbm_reqs_rcved
    dgrams_dropped_size
    dgrams_dropped_type
    dgrams_dropped_version
    dgrams_dropped_hdr
    dgrams_dropped_sid
    dgrams_dropped_other
for TCP:
    bytes_rcved
    lbm_msgs_rcved
    lbm_msgs_no_topic_rcved
    lbm_reqs_rcved
for LBTIPC:
    msgs_rcved
        bytes_rcved
        lbm_msgs_rcved
        lbm_msgs_no_topic_rcved
        lbm_reqs_rcved
for LBTRDMA:
    msgs_rcved
        bytes_rcved
        lbm_msgs_rcved
        lbm_msgs_no_topic_rcved
        lbm_reqs_rcved
*/
int
lbmmmon_rcv_format_csv_serialize(char * Destination,
                                  size_t * Size,
                                  unsigned short * ModuleID,
                                  const lbm_rcv_transport_stats_t
                                  void * FormatClientData)
{
    char work[1024];
    lbmmmon_format_csv_t      * fmt;

    if ((Destination == NULL) || (Size == NULL) || (*Size == 0) || (Statistics == NULL) ||
    {
        return (-1);
    }
    fmt = (lbmmmon_format_csv_t *) FormatClientData;

    *ModuleID = MAKE_MODULE_VERSION(LBMMON_FORMAT_CSV_VERSION_CURRENT);
    memset(work, 0, sizeof(work));
    snprintf(work, sizeof(work), "%d%c\",
                Statistics->type,
                fmt->mSeparator);
    strncat(work, Statistics->source, sizeof(work) - strlen(work) - 1);
    strncat(work, "\\", sizeof(work) - strlen(work) - 1);
    strncpy(Destination, work, *Size);
    switch (Statistics->type)
    {
        case LBM_TRANSPORT_STAT_TCP:
            snprintf(work,

```

```
        sizeof(work),
        "%c%lx%c%lx%c%lx%c%lx",
        fmt->mSeparator,
        Statistics->transport.tcp.bytes_rcved,
        fmt->mSeparator,
        Statistics->transport.tcp.lbm_msgs_rcved,
        fmt->mSeparator,
        Statistics->transport.tcp.lbm_msgs_no_topic_rcved,
        fmt->mSeparator,
        Statistics->transport.tcp.lbm_reqs_rcved);
    if (strlen(work) >= (*Size - strlen(Destination) - 1))
    {
        strncpy(ErrorString, "Destination too small for data", sizeof(ErrorString))
        return (-1);
    }
    strncat(Destination, work, *Size - strlen(Destination) - 1);
    break;

case LBM_TRANSPORT_STAT_LBTRM:
    snprintf(work,
              sizeof(work),
              "%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx",
              fmt->mSeparator,
              Statistics->transport.lbtrm.msgs_rcved,
              fmt->mSeparator,
              Statistics->transport.lbtrm.bytes_rcved,
              fmt->mSeparator,
              Statistics->transport.lbtrm.nak_pckts_sent,
              fmt->mSeparator,
              Statistics->transport.lbtrm.naks_sent,
              fmt->mSeparator,
              Statistics->transport.lbtrm.lost,
              fmt->mSeparator,
              Statistics->transport.lbtrm.ncfs_ignored,
              fmt->mSeparator,
              Statistics->transport.lbtrm.ncfs_shed,
              fmt->mSeparator,
              Statistics->transport.lbtrm.ncfs_rx_delay);
    if (strlen(work) >= (*Size - strlen(Destination) - 1))
    {
        strncpy(ErrorString, "Destination too small for data", sizeof(ErrorString))
        return (-1);
    }
    strncat(Destination, work, *Size - strlen(Destination) - 1);
    snprintf(work,
              sizeof(work),
              "%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx",
              fmt->mSeparator,
              Statistics->transport.lbtrm.ncfs_unknown,
              fmt->mSeparator,
              Statistics->transport.lbtrm.nak_stm_min,
              fmt->mSeparator,
              Statistics->transport.lbtrm.nak_stm_mean,
              fmt->mSeparator,
              Statistics->transport.lbtrm.nak_stm_max,
              fmt->mSeparator,
              Statistics->transport.lbtrm.nak_tx_min,
```

```

        fmt->mSeparator,
        Statistics->transport.lbtrm.nak_tx_mean,
        fmt->mSeparator,
        Statistics->transport.lbtrm.nak_tx_max,
        fmt->mSeparator,
        Statistics->transport.lbtrm.duplicate_data);
    if (strlen(work) >= (*Size - strlen(Destination) - 1))
    {
        strncpy(ErrorString, "Destination too small for data", sizeof(I
        return (-1);
    }
    strncat(Destination, work, *Size - strlen(Destination) - 1);
    snprintf(work,
        sizeof(work),
        "%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx",
        fmt->mSeparator,
        Statistics->transport.lbtrm.unrecovered_txw,
        fmt->mSeparator,
        Statistics->transport.lbtrm.unrecovered_tmo,
        fmt->mSeparator,
        Statistics->transport.lbtrm.lbm_msgs_rcved,
        fmt->mSeparator,
        Statistics->transport.lbtrm.lbm_msgs_no_topic_rcved,
        fmt->mSeparator,
        Statistics->transport.lbtrm.lbm_reqs_rcved,
        fmt->mSeparator,
        Statistics->transport.lbtrm.dgrams_dropped_size,
        fmt->mSeparator,
        Statistics->transport.lbtrm.dgrams_dropped_type,
        fmt->mSeparator,
        Statistics->transport.lbtrm.dgrams_dropped_version);
    if (strlen(work) >= (*Size - strlen(Destination) - 1))
    {
        strncpy(ErrorString, "Destination too small for data", sizeof(I
        return (-1);
    }
    strncat(Destination, work, *Size - strlen(Destination) - 1);
    snprintf(work,
        sizeof(work),
        "%c%lx%c%lx%c%lx",
        fmt->mSeparator,
        Statistics->transport.lbtrm.dgrams_dropped_hdr,
        fmt->mSeparator,
        Statistics->transport.lbtrm.dgrams_dropped_other,
        fmt->mSeparator,
        Statistics->transport.lbtrm.out_of_order);
    if (strlen(work) >= (*Size - strlen(Destination) - 1))
    {
        strncpy(ErrorString, "Destination too small for data", sizeof(I
        return (-1);
    }
    strncat(Destination, work, *Size - strlen(Destination) - 1);
    break;

case LBM_TRANSPORT_STAT_LBTRU:
    snprintf(work,
        sizeof(work),

```

```
%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx",
fmt->mSeparator,
Statistics->transport.lbtru.msgs_rcved,
fmt->mSeparator,
Statistics->transport.lbtru.bytes_rcved,
fmt->mSeparator,
Statistics->transport.lbtru.nak_pkts_sent,
fmt->mSeparator,
Statistics->transport.lbtru.naks_sent,
fmt->mSeparator,
Statistics->transport.lbtru.lost,
fmt->mSeparator,
Statistics->transport.lbtru.ncfs_ignored,
fmt->mSeparator,
Statistics->transport.lbtru.ncfs_shed,
fmt->mSeparator,
Statistics->transport.lbtru.ncfs_rx_delay);
if (strlen(work) >= (*Size - strlen(Destination) - 1))
{
    strncpy(ErrorString, "Destination too small for data", sizeof(ErrorString));
    return (-1);
}
strncat(Destination, work, *Size - strlen(Destination) - 1);
snprintf(work,
         sizeof(work),
         "%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx",
         fmt->mSeparator,
         Statistics->transport.lbtru.ncfs_unknown,
         fmt->mSeparator,
         Statistics->transport.lbtru.nak_stm_min,
         fmt->mSeparator,
         Statistics->transport.lbtru.nak_stm_mean,
         fmt->mSeparator,
         Statistics->transport.lbtru.nak_stm_max,
         fmt->mSeparator,
         Statistics->transport.lbtru.nak_tx_min,
         fmt->mSeparator,
         Statistics->transport.lbtru.nak_tx_mean,
         fmt->mSeparator,
         Statistics->transport.lbtru.nak_tx_max,
         fmt->mSeparator,
         Statistics->transport.lbtru.duplicate_data);
if (strlen(work) >= (*Size - strlen(Destination) - 1))
{
    strncpy(ErrorString, "Destination too small for data", sizeof(ErrorString));
    return (-1);
}
strncat(Destination, work, *Size - strlen(Destination) - 1);
snprintf(work,
         sizeof(work),
         "%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx",
         fmt->mSeparator,
         Statistics->transport.lbtru.unrecovered_txw,
         fmt->mSeparator,
         Statistics->transport.lbtru.unrecovered_tmo,
         fmt->mSeparator,
         Statistics->transport.lbtru.lbm_msgs_rcved,
```

```

        fmt->mSeparator,
        Statistics->transport.lbtru.lbm_msgs_no_topic_rcved,
        fmt->mSeparator,
        Statistics->transport.lbtru.lbm_reqs_rcved,
        fmt->mSeparator,
        Statistics->transport.lbtru.dgrams_dropped_size,
        fmt->mSeparator,
        Statistics->transport.lbtru.dgrams_dropped_type,
        fmt->mSeparator,
        Statistics->transport.lbtru.dgrams_dropped_version);
    if (strlen(work) >= (*Size - strlen(Destination) - 1))
    {
        strncpy(ErrorString, "Destination too small for data", sizeof(ErrorString));
        return (-1);
    }
    strncat(Destination, work, *Size - strlen(Destination) - 1);
    sprintf(work,
            sizeof(work),
            "%c%lx%c%lx%c%lx",
            fmt->mSeparator,
            Statistics->transport.lbtru.dgrams_dropped_hdr,
            fmt->mSeparator,
            Statistics->transport.lbtru.dgrams_dropped_sid,
            fmt->mSeparator,
            Statistics->transport.lbtru.dgrams_dropped_other);
    if (strlen(work) >= (*Size - strlen(Destination) - 1))
    {
        strncpy(ErrorString, "Destination too small for data", sizeof(ErrorString));
        return (-1);
    }
    strncat(Destination, work, *Size - strlen(Destination) - 1);
    break;

case LBM_TRANSPORT_STAT_LBTIPC:
    sprintf(work,
            sizeof(work),
            "%c%lx%c%lx%c%lx%c%lx",
            fmt->mSeparator,
            Statistics->transport.lbtipc.msgs_rcved,
            fmt->mSeparator,
            Statistics->transport.lbtipc.bytes_rcved,
            fmt->mSeparator,
            Statistics->transport.lbtipc.lbm_msgs_rcved,
            fmt->mSeparator,
            Statistics->transport.lbtipc.lbm_msgs_no_topic_rcved,
            fmt->mSeparator,
            Statistics->transport.lbtipc.lbm_reqs_rcved);
    if (strlen(work) >= (*Size - strlen(Destination) - 1))
    {
        strncpy(ErrorString, "Destination too small for data", sizeof(ErrorString));
        return (-1);
    }
    strncat(Destination, work, *Size - strlen(Destination) - 1);
    break;

case LBM_TRANSPORT_STAT_LBTRDMA:
    sprintf(work,

```

```

        sizeof(work),
        "%c%lx%c%lx%c%lx%c%lx",
        fmt->mSeparator,
        Statistics->transport.lbtrdma.msgs_rcved,
        fmt->mSeparator,
        Statistics->transport.lbtrdma.bytes_rcved,
        fmt->mSeparator,
        Statistics->transport.lbtrdma.lbm_msgs_rcved,
        fmt->mSeparator,
        Statistics->transport.lbtrdma.lbm_msgs_no_topic_rcved,
        fmt->mSeparator,
        Statistics->transport.lbtrdma.lbm_reqs_rcved);
    if (strlen(work) >= (*Size - strlen(Destination) - 1))
    {
        strncpy(ErrorString, "Destination too small for data", sizeof(ErrorString));
        return (-1);
    }
    strncat(Destination, work, *Size - strlen(Destination) - 1);
    break;
}

default:
    strncpy(ErrorString, "Unknown LBM transport type", sizeof(ErrorString));
    return (-1);
}
*Size = strlen(Destination);
return (0);
}

/*
Format of the CSV source statistics data is:
type (one of LBM_TRANSPORT_STAT_*) values
source (as a string)
actual statistics, depending on type....
for LBTRM:
    msgs_sent
    bytes_sent
    txw_msgs
    txw_bytes
    nak_pckts_rcved
    naks_rcved
    naks_ignored
    naks_shed
    naks_rx_delay_ignored
    rxs_sent
    rctlr_data_msgs
    rctlr_rx_msgs
    rx_bytes_sent
for LBTRU:
    msgs_sent
    bytes_sent
    nak_pckts_rcved
    naks_rcved
    naks_ignored
    naks_shed
    naks_rx_delay_ignored
    rxs_sent
    num_clients

```

```

        rx_bytes_sent
    for TCP:
        num_clients
        bytes_buffered
    for LBTRIPC:
        num_clients
            msgs_sent
            bytes_sent
    for LBTRDMA:
        num_clients
            msgs_sent
            bytes_sent
    */

int
lbmon_src_format_csv_serialize(char * Destination,
                                size_t * Size,
                                unsigned short * ModuleID,
                                const lbm_src_transport_stats_t
                                void * FormatClientData)
{
    char work[1024];
    lbmon_format_csv_t      * fmt;

    if ((Destination == NULL) || (Size == NULL) || (*Size == 0) || (Statistics == NULL) ||
    {
        return (-1);
    }
    fmt = (lbmon_format_csv_t *) FormatClientData;

    *ModuleID = MAKE_MODULE_VERSION(LBMMON_FORMAT_CSV_VERSION_CURRENT);
    memset(work, 0, sizeof(work));
    snprintf(work, sizeof(work), "%d%c\",
                    Statistics->type,
                    fmt->mSeparator);
    strncat(work, Statistics->source, sizeof(work) - strlen(work) - 1);
    strncat(work, "\", sizeof(work) - strlen(work) - 1);
    strncpy(Destination, work, *Size);
    switch (Statistics->type)
    {
        case LBM_TRANSPORT_STAT_TCP:
            snprintf(work,
                    sizeof(work),
                    "%c%lx%c%lx",
                    fmt->mSeparator,
                    Statistics->transport.tcp.num_clients,
                    fmt->mSeparator,
                    Statistics->transport.tcp.bytes_buffered);
            if (strlen(work) >= (*Size - strlen(Destination) - 1))
            {
                strncpy(ErrorString, "Destination too small for data", sizeof(ErrorString));
                return (-1);
            }
            strncat(Destination, work, *Size - strlen(Destination) - 1);
            break;
        case LBM_TRANSPORT_STAT_LBTRM:

```

```

snprintf(work,
          sizeof(work),
          "%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx",
          fmt->mSeparator,
          Statistics->transport.lbtrm.msgs_sent,
          fmt->mSeparator,
          Statistics->transport.lbtrm.bytes_sent,
          fmt->mSeparator,
          Statistics->transport.lbtrm.txw_msgs,
          fmt->mSeparator,
          Statistics->transport.lbtrm.txw_bytes,
          fmt->mSeparator,
          Statistics->transport.lbtrm.nak_pckts_rcved,
          fmt->mSeparator,
          Statistics->transport.lbtrm.naks_rcved,
          fmt->mSeparator,
          Statistics->transport.lbtrm.naks_ignored,
          fmt->mSeparator,
          Statistics->transport.lbtrm.naks_shed);
if (strlen(work) >= (*Size - strlen(Destination) - 1))
{
    strncpy(ErrorString, "Destination too small for data", sizeof(ErrorString));
    return (-1);
}
strncat(Destination, work, *Size - strlen(Destination) - 1);
snprintf(work,
          sizeof(work),
          "%c%lx%c%lx%c%lx%c%lx",
          fmt->mSeparator,
          Statistics->transport.lbtrm.naks_rx_delay_ignored,
          fmt->mSeparator,
          Statistics->transport.lbtrm.rxs_sent,
          fmt->mSeparator,
          Statistics->transport.lbtrm.rctlr_data_msgs,
          fmt->mSeparator,
          Statistics->transport.lbtrm.rctlr_rx_msgs,
          fmt->mSeparator,
          Statistics->transport.lbtrm.rx_bytes_sent);
if (strlen(work) >= (*Size - strlen(Destination) - 1))
{
    strncpy(ErrorString, "Destination too small for data", sizeof(ErrorString));
    return (-1);
}
strncat(Destination, work, *Size - strlen(Destination) - 1);
break;

case LBM_TRANSPORT_STAT_LBTRU:
    snprintf(work,
              sizeof(work),
              "%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx",
              fmt->mSeparator,
              Statistics->transport.lbtru.msgs_sent,
              fmt->mSeparator,
              Statistics->transport.lbtru.bytes_sent,
              fmt->mSeparator,
              Statistics->transport.lbtru.nak_pckts_rcved,
              fmt->mSeparator,

```

```

        Statistics->transport.lbtru.naks_rcved,
        fmt->mSeparator,
        Statistics->transport.lbtru.naks_ignored,
        fmt->mSeparator,
        Statistics->transport.lbtru.naks_shed,
        fmt->mSeparator,
        Statistics->transport.lbtru.naks_rx_delay_ignored,
        fmt->mSeparator,
        Statistics->transport.lbtru.rxs_sent);
    if (strlen(work) >= (*Size - strlen(Destination) - 1))
    {
        strncpy(ErrorString, "Destination too small for data", sizeof(I
        return (-1);
    }
    strncat(Destination, work, *Size - strlen(Destination) - 1);
    snprintf(work,
              sizeof(work),
              "%c%lx%c%lx",
              fmt->mSeparator,
              Statistics->transport.lbtru.num_clients,
              fmt->mSeparator,
              Statistics->transport.lbtru.rx_bytes_sent);
    if (strlen(work) >= (*Size - strlen(Destination) - 1))
    {
        return (-1);
    }
    strncat(Destination, work, *Size - strlen(Destination) - 1);
    break;

case LBM_TRANSPORT_STAT_LBTIPC:
    snprintf(work,
              sizeof(work),
              "%c%lx%c%lx%c%lx",
              fmt->mSeparator,
              Statistics->transport.lbtipc.num_clients,
              fmt->mSeparator,
              Statistics->transport.lbtipc.msgs_sent,
              fmt->mSeparator,
              Statistics->transport.lbtipc.bytes_sent);
    if (strlen(work) >= (*Size - strlen(Destination) - 1))
    {
        strncpy(ErrorString, "Destination too small for data", sizeof(I
        return (-1);
    }
    strncat(Destination, work, *Size - strlen(Destination) - 1);
    break;
case LBM_TRANSPORT_STAT_LBTRDMA:
    snprintf(work,
              sizeof(work),
              "%c%lx%c%lx%c%lx",
              fmt->mSeparator,
              Statistics->transport.lbtrdma.num_clients,
              fmt->mSeparator,
              Statistics->transport.lbtrdma.msgs_sent,
              fmt->mSeparator,
              Statistics->transport.lbtrdma.bytes_sent);
    if (strlen(work) >= (*Size - strlen(Destination) - 1))

```

```
{  
    strncpy(ErrorString, "Destination too small for data", sizeof(ErrorString))  
    return (-1);  
}  
strncat(Destination, work, *Size - strlen(Destination) - 1);  
break;  
}  
*Size = strlen(Destination);  
return (0);  
}  
  
/*  
Format of the CSV event queue statistics data is:  
data_msgs  
data_msgs_tot  
data_msgs_svc_min  
data_msgs_svc_mean  
data_msgs_svc_max  
resp_msgs  
resp_msgs_tot  
resp_msgs_svc_min  
resp_msgs_svc_mean  
resp_msgs_svc_max  
topicless_im_msgs  
topicless_im_msgs_tot  
topicless_im_msgs_svc_min  
topicless_im_msgs_svc_mean  
topicless_im_msgs_svc_max  
wrcv_msgs  
wrcv_msgs_tot  
wrcv_msgs_svc_min  
wrcv_msgs_svc_mean  
wrcv_msgs_svc_max  
io_events  
io_events_tot  
io_events_svc_min  
io_events_svc_mean  
io_events_svc_max  
timer_events  
timer_events_tot  
timer_events_svc_min  
timer_events_svc_mean  
timer_events_svc_max  
source_events  
source_events_tot  
source_events_svc_min  
source_events_svc_mean  
source_events_svc_max  
unblock_events  
unblock_events_tot  
cancel_events  
cancel_events_tot  
cancel_events_svc_min  
cancel_events_svc_mean  
cancel_events_svc_max  
context_source_events  
context_source_events_tot
```



```

Statistics->topicless_im_msgs,
fmt->mSeparator,
Statistics->topicless_im_msgs_tot,
fmt->mSeparator,
Statistics->topicless_im_msgs_svc_min,
fmt->mSeparator,
Statistics->topicless_im_msgs_svc_mean,
fmt->mSeparator,
Statistics->topicless_im_msgs_svc_max,
fmt->mSeparator);
if (strlen(work) >= (*Size - strlen(Destination) - 1))
{
    strncpy(ErrorString, "Destination too small for data", sizeof(ErrorString));
    return (-1);
}
strncat(Destination, work, *Size - strlen(Destination) - 1);
snprintf(work,
        sizeof(work),
        "%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c",
        Statistics->wrcv_msgs,
        fmt->mSeparator,
        Statistics->wrcv_msgs_tot,
        fmt->mSeparator,
        Statistics->wrcv_msgs_svc_min,
        fmt->mSeparator,
        Statistics->wrcv_msgs_svc_mean,
        fmt->mSeparator,
        Statistics->wrcv_msgs_svc_max,
        fmt->mSeparator,
        Statistics->io_events,
        fmt->mSeparator,
        Statistics->io_events_tot,
        fmt->mSeparator,
        Statistics->io_events_svc_min,
        fmt->mSeparator,
        Statistics->io_events_svc_mean,
        fmt->mSeparator,
        Statistics->io_events_svc_max,
        fmt->mSeparator,
        Statistics->timer_events,
        fmt->mSeparator,
        Statistics->timer_events_tot,
        fmt->mSeparator,
        Statistics->timer_events_svc_min,
        fmt->mSeparator,
        Statistics->timer_events_svc_mean,
        fmt->mSeparator,
        Statistics->timer_events_svc_max,
        fmt->mSeparator);
if (strlen(work) >= (*Size - strlen(Destination) - 1))
{
    strncpy(ErrorString, "Destination too small for data", sizeof(ErrorString));
    return (-1);
}
strncat(Destination, work, *Size - strlen(Destination) - 1);
snprintf(work,
        sizeof(work),

```



```

        fmt->mSeparator,
        Statistics->callback_events_svc_min,
        fmt->mSeparator,
        Statistics->callback_events_svc_mean,
        fmt->mSeparator,
        Statistics->callback_events_svc_max);
    if (strlen(work) >= (*Size - strlen(Destination) - 1))
    {
        strncpy(ErrorString, "Destination too small for data", sizeof(ErrorString));
        return (-1);
    }
    strncat(Destination, work, *Size - strlen(Destination) - 1);
    *Size = strlen(Destination);
    return (0);
}

/*
Format of the CSV context statistics data is:
tr_dgrams_sent
tr_bytes_sent
tr_dgrams_rcved
tr_bytes_rcved
tr_dgrams_dropped_ver
tr_dgrams_dropped_type
tr_dgrams_dropped_malformed
tr_dgrams_send_failed
tr_src_topics
tr_rcv_topics
tr_rcv_unresolved_topics
lbtrm_unknown_msgs_rcved
lbtru_unknown_msgs_rcved
send_blocked
send_would_block
resp_blocked
resp_would_block
uim_dup_msgs_rcved
uim_msg_no_stream_rcved
*/
int
lbmon_ctx_format_csv_serialize(char * Destination,
                                size_t * Size,
                                unsigned short * ModuleID,
                                const lbm_context_stats_t * Statistics,
                                void * FormatClientData)
{
    lbmon_format_csv_t      * fmt;
    char work[1024];

    if ((Destination == NULL) || (Size == NULL) || (*Size == 0) || (Statistics == NULL) || (FormatClientData == NULL))
    {
        return (-1);
    }
    fmt = (lbmon_format_csv_t *) FormatClientData;

    *ModuleID = MAKE_MODULE_VERSION(LBMON_FORMAT_CSV_VERSION_CURRENT);
    memset(work, 0, sizeof(work));
}

```



```

        return (0);
    }

const char *
next_csv_value(const char * String,
              char * Value,
              size_t Size,
              char Separator)
{
    const char * ptr = String;
    size_t pos;

    if ((ptr == NULL) || (Value == NULL) || (Size == 0))
    {
        return (NULL);
    }
    memset(Value, 0, Size);

    /* Skip any whitespace */
    while ((*ptr != '\0') && (*ptr != Separator) && ((*ptr == ' ') || (*ptr == '\t')))
    {
        ptr++;
    }
    pos = 0;

    if (*ptr == '\0')
    {
        return (NULL);
    }
    else if (*ptr == Separator)
    {
        ptr++;
        return (ptr);
    }
    else if ((*ptr == '\"') || (*ptr == '\''))
    {
        char quote = *ptr;
        ptr++;
        while ((*ptr != '\0') && (*ptr != quote) && (pos < (Size - 1)))
        {
            Value[pos++] = *ptr++;
        }
        /* In case we exceeded the Value size, scan for the ending quote. */
        while ((*ptr != '\0') && (*ptr != quote))
        {
            ptr++;
        }
        /* Finally, scan for the separator */
        while ((*ptr != '\0') && (*ptr != Separator))
        {
            ptr++;
        }
    }
    else
    {
        /* Copy into Value */
        while ((*ptr != '\0') && (*ptr != Separator) && (pos < (Size - 1)))

```

```

{
    Value[pos++] = *ptr++;
}
/* In case we exceeded the Value size, scan for the separator. */
while ((*ptr != '\0') && (*ptr != Separator))
{
    ptr++;
}
/* If we're at the separator, advance the pointer */
if (*ptr == Separator)
{
    ptr++;
}
return (ptr);
}

static lbm_ulong_t
convert_value(const char * Buffer)
{
    lbm_ulong_t value = 0;
    const char * ptr = Buffer;

    while (1)
    {
        errno = 0;
        value = strtoul(ptr, NULL, 16);
        if ((value == ULONG_MAX) && (errno == ERANGE))
        {
            ptr++;
        }
        else
        {
            return (value);
        }
    }
}

/*********************************************
/* A note to maintainers:
*/
/*
* Ideally, the code to deserialize statistics would be completely generic. Instead of se-
* parsing loops for each of n message types, a single function could parse the string and
* the values into the structure (given the appropriate pointers). Access to the statistic
* structure would also be generic, casting a pointer to the actual structure into a char
* then indexing using the field offset arrays (below) to locate the correct position for
* field within the structure.
*
* That is, as long as the type of each field in every statistics structure is the same.
* Which it currently is... and probably will remain so. But there's no guarantee that it
* _will_ remain so. So better to bite the bullet now, and make the possibility of non-
* homogeneous structures simple to implement.
*/
********************************************

static const size_t csv_rcv_tcp_stat_offset_v1[] =
{
    offsetof(lbm_rcv_transport_stats_tcp_t, bytes_rcved),

```

```

        offsetof(lbm_rcv_transport_stats_tcp_t, lbm_msgs_rcved),
        offsetof(lbm_rcv_transport_stats_tcp_t, lbm_msgs_no_topic_rcved),
        offsetof(lbm_rcv_transport_stats_tcp_t, lbm_reqs_rcved)
    };
#define csv_rcv_tcp_stat_offset_v2 csv_rcv_tcp_stat_offset_v1
#define csv_rcv_tcp_stat_offset_v3 csv_rcv_tcp_stat_offset_v2
#define csv_rcv_tcp_stat_offset_v4 csv_rcv_tcp_stat_offset_v3
static const lbmon_csv_layout_t csv_rcv_tcp_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT+1] =
{
    { NULL, 0 },
    { csv_rcv_tcp_stat_offset_v1, sizeof(csv_rcv_tcp_stat_offset_v1)/sizeof(csv_rcv_tcp_stat_offset_v1) },
    { csv_rcv_tcp_stat_offset_v2, sizeof(csv_rcv_tcp_stat_offset_v2)/sizeof(csv_rcv_tcp_stat_offset_v2) },
    { csv_rcv_tcp_stat_offset_v3, sizeof(csv_rcv_tcp_stat_offset_v3)/sizeof(csv_rcv_tcp_stat_offset_v3) },
    { csv_rcv_tcp_stat_offset_v4, sizeof(csv_rcv_tcp_stat_offset_v4)/sizeof(csv_rcv_tcp_stat_offset_v4) }
};

static const size_t csv_rcv_lbtrm_stat_offset_v1[] =
{
    offsetof(lbm_rcv_transport_stats_lbtrm_t, msgs_rcved),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, bytes_rcved),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, nak_pkts_sent),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, naks_sent),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, lost),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, ncfs_ignored),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, ncfs_shed),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, ncfs_rx_delay),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, ncfs_unknown),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, nak_stm_min),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, nak_stm_mean),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, nak_stm_max),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, nak_tx_min),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, nak_tx_mean),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, nak_tx_max),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, duplicate_data),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, unrecovered_txw),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, unrecovered_tmo),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, lbm_msgs_rcved),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, lbm_msgs_no_topic_rcved),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, lbm_reqs_rcved)
};
#define csv_rcv_lbtrm_stat_offset_v2 csv_rcv_lbtrm_stat_offset_v1
static const size_t csv_rcv_lbtrm_stat_offset_v3[] =
{
    offsetof(lbm_rcv_transport_stats_lbtrm_t, msgs_rcved),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, bytes_rcved),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, nak_pkts_sent),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, naks_sent),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, lost),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, ncfs_ignored),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, ncfs_shed),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, ncfs_rx_delay),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, ncfs_unknown),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, nak_stm_min),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, nak_stm_mean),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, nak_stm_max),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, nak_tx_min),
    offsetof(lbm_rcv_transport_stats_lbtrm_t, nak_tx_mean),

```

```

        offsetof(lbm_rcv_transport_stats_lbtrm_t, nak_tx_max),
        offsetof(lbm_rcv_transport_stats_lbtrm_t, duplicate_data),
        offsetof(lbm_rcv_transport_stats_lbtrm_t, unrecovered_txw),
        offsetof(lbm_rcv_transport_stats_lbtrm_t, unrecovered_tmo),
        offsetof(lbm_rcv_transport_stats_lbtrm_t, lbm_msgs_rcved),
        offsetof(lbm_rcv_transport_stats_lbtrm_t, lbm_msgs_no_topic_rcved),
        offsetof(lbm_rcv_transport_stats_lbtrm_t, lbm_reqs_rcved),
        offsetof(lbm_rcv_transport_stats_lbtrm_t, dgrams_dropped_size),
        offsetof(lbm_rcv_transport_stats_lbtrm_t, dgrams_dropped_type),
        offsetof(lbm_rcv_transport_stats_lbtrm_t, dgrams_dropped_version),
        offsetof(lbm_rcv_transport_stats_lbtrm_t, dgrams_dropped_hdr),
        offsetof(lbm_rcv_transport_stats_lbtrm_t, dgrams_dropped_other),
        offsetof(lbm_rcv_transport_stats_lbtrm_t, out_of_order)
    };
#define csv_rcv_lbtrm_stat_offset_v4 csv_rcv_lbtrm_stat_offset_v3
static const lbmmmon_csv_layout_t csv_rcv_lbtrm_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT+1]
{
    { NULL, 0 },
    { csv_rcv_lbtrm_stat_offset_v1, sizeof(csv_rcv_lbtrm_stat_offset_v1)/sizeof(csv_rcv_lbtrm_stat_offset_v3) },
    { csv_rcv_lbtrm_stat_offset_v2, sizeof(csv_rcv_lbtrm_stat_offset_v2)/sizeof(csv_rcv_lbtrm_stat_offset_v3) },
    { csv_rcv_lbtrm_stat_offset_v3, sizeof(csv_rcv_lbtrm_stat_offset_v3)/sizeof(csv_rcv_lbtrm_stat_offset_v3) },
    { csv_rcv_lbtrm_stat_offset_v4, sizeof(csv_rcv_lbtrm_stat_offset_v4)/sizeof(csv_rcv_lbtrm_stat_offset_v3) }
};

static const size_t csv_rcv_lbtru_stat_offset_v1[] =
{
    offsetof(lbm_rcv_transport_stats_lbtru_t, msgs_rcved),
    offsetof(lbm_rcv_transport_stats_lbtru_t, bytes_rcved),
    offsetof(lbm_rcv_transport_stats_lbtru_t, nak_pkts_sent),
    offsetof(lbm_rcv_transport_stats_lbtru_t, naks_sent),
    offsetof(lbm_rcv_transport_stats_lbtru_t, lost),
    offsetof(lbm_rcv_transport_stats_lbtru_t, ncfs_ignored),
    offsetof(lbm_rcv_transport_stats_lbtru_t, ncfs_shed),
    offsetof(lbm_rcv_transport_stats_lbtru_t, ncfs_rx_delay),
    offsetof(lbm_rcv_transport_stats_lbtru_t, ncfs_unknown),
    offsetof(lbm_rcv_transport_stats_lbtru_t, nak_stm_min),
    offsetof(lbm_rcv_transport_stats_lbtru_t, nak_stm_mean),
    offsetof(lbm_rcv_transport_stats_lbtru_t, nak_stm_max),
    offsetof(lbm_rcv_transport_stats_lbtru_t, nak_tx_min),
    offsetof(lbm_rcv_transport_stats_lbtru_t, nak_tx_mean),
    offsetof(lbm_rcv_transport_stats_lbtru_t, nak_tx_max),
    offsetof(lbm_rcv_transport_stats_lbtru_t, duplicate_data),
    offsetof(lbm_rcv_transport_stats_lbtru_t, unrecovered_txw),
    offsetof(lbm_rcv_transport_stats_lbtru_t, unrecovered_tmo),
    offsetof(lbm_rcv_transport_stats_lbtru_t, lbm_msgs_rcved),
    offsetof(lbm_rcv_transport_stats_lbtru_t, lbm_msgs_no_topic_rcved),
    offsetof(lbm_rcv_transport_stats_lbtru_t, lbm_reqs_rcved)
};
#define csv_rcv_lbtru_stat_offset_v2 csv_rcv_lbtru_stat_offset_v1
static const size_t csv_rcv_lbtru_stat_offset_v3[] =
{
    offsetof(lbm_rcv_transport_stats_lbtru_t, msgs_rcved),
    offsetof(lbm_rcv_transport_stats_lbtru_t, bytes_rcved),
    offsetof(lbm_rcv_transport_stats_lbtru_t, nak_pkts_sent),
    offsetof(lbm_rcv_transport_stats_lbtru_t, naks_sent),
    offsetof(lbm_rcv_transport_stats_lbtru_t, lost),
    offsetof(lbm_rcv_transport_stats_lbtru_t, ncfs_ignored),

```

```

        offsetof(lbm_rcv_transport_stats_lbtru_t, ncfs_shed),
        offsetof(lbm_rcv_transport_stats_lbtru_t, ncfs_rx_delay),
        offsetof(lbm_rcv_transport_stats_lbtru_t, ncfs_unknown),
        offsetof(lbm_rcv_transport_stats_lbtru_t, nak_stm_min),
        offsetof(lbm_rcv_transport_stats_lbtru_t, nak_stm_mean),
        offsetof(lbm_rcv_transport_stats_lbtru_t, nak_stm_max),
        offsetof(lbm_rcv_transport_stats_lbtru_t, nak_tx_min),
        offsetof(lbm_rcv_transport_stats_lbtru_t, nak_tx_mean),
        offsetof(lbm_rcv_transport_stats_lbtru_t, nak_tx_max),
        offsetof(lbm_rcv_transport_stats_lbtru_t, duplicate_data),
        offsetof(lbm_rcv_transport_stats_lbtru_t, unrecovered_txw),
        offsetof(lbm_rcv_transport_stats_lbtru_t, unrecovered_tmo),
        offsetof(lbm_rcv_transport_stats_lbtru_t, lbm_msgs_rcved),
        offsetof(lbm_rcv_transport_stats_lbtru_t, lbm_msgs_no_topic_rcved),
        offsetof(lbm_rcv_transport_stats_lbtru_t, lbm_reqs_rcved),
        offsetof(lbm_rcv_transport_stats_lbtru_t, dgrams_dropped_size),
        offsetof(lbm_rcv_transport_stats_lbtru_t, dgrams_dropped_type),
        offsetof(lbm_rcv_transport_stats_lbtru_t, dgrams_dropped_version),
        offsetof(lbm_rcv_transport_stats_lbtru_t, dgrams_dropped_hdr),
        offsetof(lbm_rcv_transport_stats_lbtru_t, dgrams_dropped_sid),
        offsetof(lbm_rcv_transport_stats_lbtru_t, dgrams_dropped_other)
    };
#define csv_rcv_lbtru_stat_offset_v4 csv_rcv_lbtru_stat_offset_v3
static const lbmon_csv_layout_t csv_rcv_lbtru_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT+1] =
{
    { NULL, 0 },
    { csv_rcv_lbtru_stat_offset_v1, sizeof(csv_rcv_lbtru_stat_offset_v1)/sizeof(csv_rcv_lbtru_stat_offset_v1) },
    { csv_rcv_lbtru_stat_offset_v2, sizeof(csv_rcv_lbtru_stat_offset_v2)/sizeof(csv_rcv_lbtru_stat_offset_v2) },
    { csv_rcv_lbtru_stat_offset_v3, sizeof(csv_rcv_lbtru_stat_offset_v3)/sizeof(csv_rcv_lbtru_stat_offset_v3) },
    { csv_rcv_lbtru_stat_offset_v4, sizeof(csv_rcv_lbtru_stat_offset_v4)/sizeof(csv_rcv_lbtru_stat_offset_v4) }
};

static const size_t csv_rcv_lbtipc_stat_offset_v2[] =
{
    offsetof(lbm_rcv_transport_stats_lbtipc_t, msgs_rcved),
    offsetof(lbm_rcv_transport_stats_lbtipc_t, bytes_rcved),
    offsetof(lbm_rcv_transport_stats_lbtipc_t, lbm_msgs_rcved),
    offsetof(lbm_rcv_transport_stats_lbtipc_t, lbm_msgs_no_topic_rcved),
    offsetof(lbm_rcv_transport_stats_lbtipc_t, lbm_reqs_rcved)
};
#define csv_rcv_lbtipc_stat_offset_v3 csv_rcv_lbtipc_stat_offset_v2
#define csv_rcv_lbtipc_stat_offset_v4 csv_rcv_lbtipc_stat_offset_v3
static const lbmon_csv_layout_t csv_rcv_lbtipc_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT+1] =
{
    { NULL, 0 },
    { NULL, 0 },
    { csv_rcv_lbtipc_stat_offset_v2, sizeof(csv_rcv_lbtipc_stat_offset_v2)/sizeof(csv_rcv_lbtipc_stat_offset_v2) },
    { csv_rcv_lbtipc_stat_offset_v3, sizeof(csv_rcv_lbtipc_stat_offset_v3)/sizeof(csv_rcv_lbtipc_stat_offset_v3) },
    { csv_rcv_lbtipc_stat_offset_v4, sizeof(csv_rcv_lbtipc_stat_offset_v4)/sizeof(csv_rcv_lbtipc_stat_offset_v4) }
};

static const size_t csv_rcv_lbtrdma_stat_offset_v2[] =
{
    offsetof(lbm_rcv_transport_stats_lbtrdma_t, msgs_rcved),
    offsetof(lbm_rcv_transport_stats_lbtrdma_t, bytes_rcved),
    offsetof(lbm_rcv_transport_stats_lbtrdma_t, lbm_msgs_rcved),
    offsetof(lbm_rcv_transport_stats_lbtrdma_t, lbm_msgs_no_topic_rcved),
    offsetof(lbm_rcv_transport_stats_lbtrdma_t, lbm_reqs_rcved)
};

```

```

        offsetof(lbm_rcv_transport_stats_lbtrdma_t, lbm_reqs_rcved)
};

#define csv_rcv_lbtrdma_stat_offset_v3 csv_rcv_lbtrdma_stat_offset_v2
#define csv_rcv_lbtrdma_stat_offset_v4 csv_rcv_lbtrdma_stat_offset_v3
static const lbmmon_csv_layout_t csv_rcv_lbtrdma_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT]
{
    { NULL, 0 },
    { NULL, 0 },
    { csv_rcv_lbtrdma_stat_offset_v2, sizeof(csv_rcv_lbtrdma_stat_offset_v2)/sizeof(csv_rcv_lbtrdma_stat_offset_v2) },
    { csv_rcv_lbtrdma_stat_offset_v3, sizeof(csv_rcv_lbtrdma_stat_offset_v3)/sizeof(csv_rcv_lbtrdma_stat_offset_v3) },
    { csv_rcv_lbtrdma_stat_offset_v4, sizeof(csv_rcv_lbtrdma_stat_offset_v4)/sizeof(csv_rcv_lbtrdma_stat_offset_v4) }
};

int
lbmmon_rcv_format_csv_deserialize(lbm_rcv_transport_stats_t * Statistics,
                                    const char * Source,
                                    size_t Length,
                                    unsigned short ModuleID,
                                    void * FormatClientData)
{
    const char * ptr;
    char value[1024];
    lbmmon_format_csv_t * fmt;
    size_t idx;
    unsigned char modid;
    unsigned char modver;
    const size_t * stat_layout = NULL;
    size_t stat_count = 0;

    if ((Statistics == NULL) || (Source == NULL) || (*Source == '\0') || (Length == 0) ||
    {
        strncpy(ErrorString, "Invalid parameter", sizeof(ErrorString));
        return (-1);
    }
    fmt = (lbmmon_format_csv_t *) FormatClientData;
    modid = MODULE_ID(ModuleID);
    modver = MODULE_VERSION(ModuleID);
    if (modid != LBMMON_FORMAT_CSV_MODULE_ID)
    {
        strncpy(ErrorString, "Invalid module ID", sizeof(ErrorString));
        return (-1);
    }

    if (fmt->mBuffer == NULL)
    {
        fmt->mBufferSize = 1024;
        fmt->mBuffer = malloc(fmt->mBufferSize);
    }
    if (Length >= fmt->mBufferSize)
    {
        fmt->mBufferSize = 2 * Length;
        free(fmt->mBuffer);
        fmt->mBuffer = malloc(fmt->mBufferSize);
    }
    memset(fmt->mBuffer, 0, fmt->mBufferSize);
    memcpy(fmt->mBuffer, Source, Length);
    ptr = fmt->mBuffer;
}

```

```

ptr = next_csv_value(ptr, value, sizeof(value), fmt->mSeparator);
if (ptr == NULL)
{
    strncpy(ErrorString, "No type field found", sizeof(ErrorString));
    return (-1);
}
Statistics->type = atoi(value);
ptr = next_csv_value(ptr, Statistics->source, sizeof(Statistics->source), fmt->mSeparator);
if (ptr == NULL)
{
    strncpy(ErrorString, "No source field found", sizeof(ErrorString));
    return (-1);
}
switch (Statistics->type)
{
    case LBM_TRANSPORT_STAT_TCP:
        if (modver >= LBMMON_FORMAT_CSV_VERSION_CURRENT)
        {
            stat_layout = csv_rcv_tcp_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].layout;
            stat_count = csv_rcv_tcp_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].count;
        }
        else
        {
            stat_layout = csv_rcv_tcp_stat_layout[modver].layout;
            stat_count = csv_rcv_tcp_stat_layout[modver].count;
        }
        memset((void *) &(Statistics->transport.tcp), 0, sizeof(lbm_rcv_transport_stats_t));
        for (idx = 0; idx < stat_count; ++idx)
        {
            ptr = next_csv_value(ptr, value, sizeof(value), fmt->mSeparator);
            if (ptr == NULL)
            {
                strncpy(ErrorString, "Data contains too few fields", sizeof(ErrorString));
                return (-1);
            }
            *((lbm_ulong_t *) (((unsigned char *)&(Statistics->transport.tcp)) + stat_l
        }
        break;

    case LBM_TRANSPORT_STAT_LBTRM:
        if (modver >= LBMMON_FORMAT_CSV_VERSION_CURRENT)
        {
            stat_layout = csv_rcv_lbtrm_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].layout;
            stat_count = csv_rcv_lbtrm_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].count;
        }
        else
        {
            stat_layout = csv_rcv_lbtrm_stat_layout[modver].layout;
            stat_count = csv_rcv_lbtrm_stat_layout[modver].count;
        }
        memset((void *) &(Statistics->transport.lbtrm), 0, sizeof(lbm_rcv_transport_stats_t));
        for (idx = 0; idx < stat_count; ++idx)
        {
            ptr = next_csv_value(ptr, value, sizeof(value), fmt->mSeparator);
            if (ptr == NULL)
            {

```

```

/* Due to an ambiguous case with version 3 lbtrm rcv st
 * older releases of lbm may have 26 fields and newer
 * See bug 5002 for more information.
 * For version 3, we will not consider it an error if t
 */
if(modver == MODULE_VERSION(3) && idx == 26) {
    return 0;
}

strncpy(ErrorString, "Data contains too few fields", s
return (-1);
}
*((lbum_ulong_t *)(((unsigned char *)&(Statistics->transport.lbtr
}
break;

case LBM_TRANSPORT_STAT_LBTRU:
if (modver >= LBMMON_FORMAT_CSV_VERSION_CURRENT)
{
    stat_layout = csv_rcv_lbtru_stat_layout[LBMMON_FORMAT_CSV_VERS
    stat_count = csv_rcv_lbtru_stat_layout[LBMMON_FORMAT_CSV_VERS
}
else
{
    stat_layout = csv_rcv_lbtru_stat_layout[modver].layout;
    stat_count = csv_rcv_lbtru_stat_layout[modver].count;
}
memset((void *) &(Statistics->transport.lbtr), 0, sizeof(lbm_rcv_trans
for (idx = 0; idx < stat_count; ++idx)
{
    ptr = next_csv_value(ptr, value, sizeof(value), fmt->mSeparator)
    if (ptr == NULL)
    {
        strncpy(ErrorString, "Data contains too few fields", s
        return (-1);
    }
    *((lbum_ulong_t *)(((unsigned char *)&(Statistics->transport.lbtr
}
break;

case LBM_TRANSPORT_STAT_LBTIPC:
if (modver >= LBMMON_FORMAT_CSV_VERSION_CURRENT)
{
    stat_layout = csv_rcv_lbtipc_stat_layout[LBMMON_FORMAT_CSV_VERS
    stat_count = csv_rcv_lbtipc_stat_layout[LBMMON_FORMAT_CSV_VERS
}
else
{
    stat_layout = csv_rcv_lbtipc_stat_layout[modver].layout;
    stat_count = csv_rcv_lbtipc_stat_layout[modver].count;
}
memset((void *) &(Statistics->transport.lbtipc), 0, sizeof(lbm_rcv_trans
for (idx = 0; idx < stat_count; ++idx)
{
    ptr = next_csv_value(ptr, value, sizeof(value), fmt->mSeparator)
    if (ptr == NULL)
    {

```

```

        strncpy(ErrorString, "Data contains too few fields", sizeof(ErrorString));
        return (-1);
    }
    *((lbm_ulong_t *)(((unsigned char *)&(Statistics->transport.lbtipc)) + stat_offset_v1));
}
break;

case LBM_TRANSPORT_STAT_LBTRDMA:
    if (modver >= LBMMON_FORMAT_CSV_VERSION_CURRENT)
    {
        stat_layout = csv_rcv_lbtrdma_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT];
        stat_count = csv_rcv_lbtrdma_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT];
    }
    else
    {
        stat_layout = csv_rcv_lbtrdma_stat_layout[modver].layout;
        stat_count = csv_rcv_lbtrdma_stat_layout[modver].count;
    }
    memset((void *)&(Statistics->transport.lbtrdma), 0, sizeof(lbm_rcv_transport_stat));
    for (idx = 0; idx < stat_count; ++idx)
    {
        ptr = next_csv_value(ptr, value, sizeof(value), fmt->mSeparator);
        if (ptr == NULL)
        {
            strncpy(ErrorString, "Data contains too few fields", sizeof(ErrorString));
            return (-1);
        }
        *((lbm_ulong_t *)(((unsigned char *)&(Statistics->transport.lbtrdma)) + stat_offset_v1));
    }
    break;

default:
    strncpy(ErrorString, "Invalid LBM transport type", sizeof(ErrorString));
    return (-1);
}
return (0);
}

static size_t csv_src_tcp_stat_offset_v1[] =
{
    offsetof(lbm_src_transport_stats_tcp_t, num_clients),
    offsetof(lbm_src_transport_stats_tcp_t, bytes_buffered)
};
#define csv_src_tcp_stat_offset_v2 csv_src_tcp_stat_offset_v1
#define csv_src_tcp_stat_offset_v3 csv_src_tcp_stat_offset_v2
#define csv_src_tcp_stat_offset_v4 csv_src_tcp_stat_offset_v3
static const lbmmmon_csv_layout_t csv_src_tcp_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT+1] =
{
    { NULL, 0 },
    { csv_src_tcp_stat_offset_v1, sizeof(csv_src_tcp_stat_offset_v1)/sizeof(csv_src_tcp_stat_offset_v1) },
    { csv_src_tcp_stat_offset_v2, sizeof(csv_src_tcp_stat_offset_v2)/sizeof(csv_src_tcp_stat_offset_v2) },
    { csv_src_tcp_stat_offset_v3, sizeof(csv_src_tcp_stat_offset_v3)/sizeof(csv_src_tcp_stat_offset_v3) },
    { csv_src_tcp_stat_offset_v4, sizeof(csv_src_tcp_stat_offset_v4)/sizeof(csv_src_tcp_stat_offset_v4) }
};

static size_t csv_src_lbtrm_stat_offset_v1[] =
{

```

```

        offsetof(lbm_src_transport_stats_lbtrm_t, msgs_sent),
        offsetof(lbm_src_transport_stats_lbtrm_t, bytes_sent),
        offsetof(lbm_src_transport_stats_lbtrm_t, txw_msgs),
        offsetof(lbm_src_transport_stats_lbtrm_t, txw_bytes),
        offsetof(lbm_src_transport_stats_lbtrm_t, nak_pkts_rcved),
        offsetof(lbm_src_transport_stats_lbtrm_t, naks_rcved),
        offsetof(lbm_src_transport_stats_lbtrm_t, naks_ignored),
        offsetof(lbm_src_transport_stats_lbtrm_t, naks_shed),
        offsetof(lbm_src_transport_stats_lbtrm_t, naks_rx_delay_ignored),
        offsetof(lbm_src_transport_stats_lbtrm_t, rxs_sent),
        offsetof(lbm_src_transport_stats_lbtrm_t, rctlr_data_msgs),
        offsetof(lbm_src_transport_stats_lbtrm_t, rctlr_rx_msgs)
    };
static size_t csv_src_lbtrm_stat_offset_v2[] =
{
    offsetof(lbm_src_transport_stats_lbtrm_t, msgs_sent),
    offsetof(lbm_src_transport_stats_lbtrm_t, bytes_sent),
    offsetof(lbm_src_transport_stats_lbtrm_t, txw_msgs),
    offsetof(lbm_src_transport_stats_lbtrm_t, txw_bytes),
    offsetof(lbm_src_transport_stats_lbtrm_t, nak_pkts_rcved),
    offsetof(lbm_src_transport_stats_lbtrm_t, naks_rcved),
    offsetof(lbm_src_transport_stats_lbtrm_t, naks_ignored),
    offsetof(lbm_src_transport_stats_lbtrm_t, naks_shed),
    offsetof(lbm_src_transport_stats_lbtrm_t, naks_rx_delay_ignored),
    offsetof(lbm_src_transport_stats_lbtrm_t, rxs_sent),
    offsetof(lbm_src_transport_stats_lbtrm_t, rctlr_data_msgs),
    offsetof(lbm_src_transport_stats_lbtrm_t, rctlr_rx_msgs),
    offsetof(lbm_src_transport_stats_lbtrm_t, rx_bytes_sent)
};
#define csv_src_lbtrm_stat_offset_v3 csv_src_lbtrm_stat_offset_v2
#define csv_src_lbtrm_stat_offset_v4 csv_src_lbtrm_stat_offset_v3
static const lbmmmon_csv_layout_t csv_src_lbtrm_stat_layout [LBMMON_FORMAT_CSV_VERSION_CURRENT+1] =
{
    { NULL, 0 },
    { csv_src_lbtrm_stat_offset_v1, sizeof(csv_src_lbtrm_stat_offset_v1)/sizeof(csv_src_lbtrm_stat_offset_v1) },
    { csv_src_lbtrm_stat_offset_v2, sizeof(csv_src_lbtrm_stat_offset_v2)/sizeof(csv_src_lbtrm_stat_offset_v2) },
    { csv_src_lbtrm_stat_offset_v3, sizeof(csv_src_lbtrm_stat_offset_v3)/sizeof(csv_src_lbtrm_stat_offset_v3) },
    { csv_src_lbtrm_stat_offset_v4, sizeof(csv_src_lbtrm_stat_offset_v4)/sizeof(csv_src_lbtrm_stat_offset_v4) }
};

static size_t csv_src_lbtru_stat_offset_v1[] =
{
    offsetof(lbm_src_transport_stats_lbtru_t, msgs_sent),
    offsetof(lbm_src_transport_stats_lbtru_t, bytes_sent),
    offsetof(lbm_src_transport_stats_lbtru_t, nak_pkts_rcved),
    offsetof(lbm_src_transport_stats_lbtru_t, naks_rcved),
    offsetof(lbm_src_transport_stats_lbtru_t, naks_ignored),
    offsetof(lbm_src_transport_stats_lbtru_t, naks_shed),
    offsetof(lbm_src_transport_stats_lbtru_t, naks_rx_delay_ignored),
    offsetof(lbm_src_transport_stats_lbtru_t, rxs_sent),
    offsetof(lbm_src_transport_stats_lbtru_t, num_clients)
};
static size_t csv_src_lbtru_stat_offset_v2[] =
{
    offsetof(lbm_src_transport_stats_lbtru_t, msgs_sent),
    offsetof(lbm_src_transport_stats_lbtru_t, bytes_sent),
    offsetof(lbm_src_transport_stats_lbtru_t, nak_pkts_rcved),

```

```

        offsetof(lbm_src_transport_stats_lbtru_t, naks_rcved),
        offsetof(lbm_src_transport_stats_lbtru_t, naks_ignored),
        offsetof(lbm_src_transport_stats_lbtru_t, naks_shed),
        offsetof(lbm_src_transport_stats_lbtru_t, naks_rx_delay_ignored),
        offsetof(lbm_src_transport_stats_lbtru_t, rxs_sent),
        offsetof(lbm_src_transport_stats_lbtru_t, num_clients),
        offsetof(lbm_src_transport_stats_lbtru_t, rx_bytes_sent)
    };
#define csv_src_lbtru_stat_offset_v3 csv_src_lbtru_stat_offset_v2
#define csv_src_lbtru_stat_offset_v4 csv_src_lbtru_stat_offset_v3
static const lbmon_csv_layout_t csv_src_lbtru_stat_layout[LBMON_FORMAT_CSV_VERSION+1] =
{
    { NULL, 0 },
    { csv_src_lbtru_stat_offset_v1, sizeof(csv_src_lbtru_stat_offset_v1)/sizeof(csv_src_lbtru_stat_offset_v2) },
    { csv_src_lbtru_stat_offset_v2, sizeof(csv_src_lbtru_stat_offset_v2)/sizeof(csv_src_lbtru_stat_offset_v3) },
    { csv_src_lbtru_stat_offset_v3, sizeof(csv_src_lbtru_stat_offset_v3)/sizeof(csv_src_lbtru_stat_offset_v4) },
    { csv_src_lbtru_stat_offset_v4, sizeof(csv_src_lbtru_stat_offset_v4)/sizeof(csv_src_lbtru_stat_offset_v5) }
};

static size_t csv_src_lbtipc_stat_offset_v2[] =
{
    offsetof(lbm_src_transport_stats_lbtipc_t, num_clients),
    offsetof(lbm_src_transport_stats_lbtipc_t, msgs_sent),
    offsetof(lbm_src_transport_stats_lbtipc_t, bytes_sent)
};
#define csv_src_lbtipc_stat_offset_v3 csv_src_lbtipc_stat_offset_v2
#define csv_src_lbtipc_stat_offset_v4 csv_src_lbtipc_stat_offset_v3
static const lbmon_csv_layout_t csv_src_lbtipc_stat_layout[LBMON_FORMAT_CSV_VERSION+1] =
{
    { NULL, 0 },
    { NULL, 0 },
    { csv_src_lbtipc_stat_offset_v2, sizeof(csv_src_lbtipc_stat_offset_v2)/sizeof(csv_src_lbtipc_stat_offset_v3) },
    { csv_src_lbtipc_stat_offset_v3, sizeof(csv_src_lbtipc_stat_offset_v3)/sizeof(csv_src_lbtipc_stat_offset_v4) },
    { csv_src_lbtipc_stat_offset_v4, sizeof(csv_src_lbtipc_stat_offset_v4)/sizeof(csv_src_lbtipc_stat_offset_v5) }
};

static size_t csv_src_lbtrdma_stat_offset_v2[] =
{
    offsetof(lbm_src_transport_stats_lbtrdma_t, num_clients),
    offsetof(lbm_src_transport_stats_lbtrdma_t, msgs_sent),
    offsetof(lbm_src_transport_stats_lbtrdma_t, bytes_sent)
};
#define csv_src_lbtrdma_stat_offset_v3 csv_src_lbtrdma_stat_offset_v2
#define csv_src_lbtrdma_stat_offset_v4 csv_src_lbtrdma_stat_offset_v3
static const lbmon_csv_layout_t csv_src_lbtrdma_stat_layout[LBMON_FORMAT_CSV_VERSION+1] =
{
    { NULL, 0 },
    { NULL, 0 },
    { csv_src_lbtrdma_stat_offset_v2, sizeof(csv_src_lbtrdma_stat_offset_v2)/sizeof(csv_src_lbtrdma_stat_offset_v3) },
    { csv_src_lbtrdma_stat_offset_v3, sizeof(csv_src_lbtrdma_stat_offset_v3)/sizeof(csv_src_lbtrdma_stat_offset_v4) },
    { csv_src_lbtrdma_stat_offset_v4, sizeof(csv_src_lbtrdma_stat_offset_v4)/sizeof(csv_src_lbtrdma_stat_offset_v5) }
};

int
lbmon_src_format_csv_deserialize(lbm_src_transport_stats_t * Statistics,
                                 const char * Source,
                                 size_t Length,

```

```

        unsigned short ModuleID,
        void * FormatClientData)
{
    const char * ptr;
    char value[1024];
    lbmmon_format_csv_t * fmt;
    size_t idx;
    unsigned char modid;
    unsigned char modver;
    const size_t * stat_layout = NULL;
    size_t stat_count = 0;

    if ((Statistics == NULL) || (Source == NULL) || (*Source == '\0') || (Length == 0) ||
    {
        strncpy(ErrorString, "Invalid parameter", sizeof(ErrorString));
        return (-1);
    }
    fmt = (lbmmon_format_csv_t *) FormatClientData;

    modid = MODULE_ID(ModuleID);
    modver = MODULE_VERSION(ModuleID);
    if (modid != LBMMON_FORMAT_CSV_MODULE_ID)
    {
        strncpy(ErrorString, "Invalid module ID", sizeof(ErrorString));
        return (-1);
    }

    if (fmt->mBuffer == NULL)
    {
        fmt->mBufferSize = 1024;
        fmt->mBuffer = malloc(fmt->mBufferSize);
    }
    if (Length >= fmt->mBufferSize)
    {
        fmt->mBufferSize = 2 * Length;
        free(fmt->mBuffer);
        fmt->mBuffer = malloc(fmt->mBufferSize);
    }
    memset(fmt->mBuffer, 0, fmt->mBufferSize);
    memcpy(fmt->mBuffer, Source, Length);
    ptr = fmt->mBuffer;
    ptr = next_csv_value(ptr, value, sizeof(value), fmt->mSeparator);
    if (ptr == NULL)
    {
        strncpy(ErrorString, "No type field found", sizeof(ErrorString));
        return (-1);
    }
    Statistics->type = atoi(value);
    ptr = next_csv_value(ptr, Statistics->source, sizeof(Statistics->source), fmt->mSeparator);
    if (ptr == NULL)
    {
        strncpy(ErrorString, "No source field found", sizeof(ErrorString));
        return (-1);
    }
    switch (Statistics->type)
    {
        case LBM_TRANSPORT_STAT_TCP:

```

```

        if (modver >= LBMMON_FORMAT_CSV_VERSION_CURRENT)
        {
            stat_layout = csv_src_tcp_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].layout;
            stat_count = csv_src_tcp_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].count;
        }
        else
        {
            stat_layout = csv_src_tcp_stat_layout[modver].layout;
            stat_count = csv_src_tcp_stat_layout[modver].count;
        }
        memset((void *) &(Statistics->transport.tcp), 0, sizeof(lbm_src_transport_stats_t));
        for (idx = 0; idx < stat_count; ++idx)
        {
            ptr = next_csv_value(ptr, value, sizeof(value), fmt->mSeparator);
            if (ptr == NULL)
            {
                strncpy(ErrorString, "Data contains too few fields", sizeof(ErrorString));
                return (-1);
            }
            *((lbum_ulong_t *)(((unsigned char *)&(Statistics->transport.tcp)) + stat_layout + idx));
        }
        break;
    }

    case LBM_TRANSPORT_STAT_LBTRM:
        if (modver >= LBMMON_FORMAT_CSV_VERSION_CURRENT)
        {
            stat_layout = csv_src_lbtrm_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].layout;
            stat_count = csv_src_lbtrm_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].count;
        }
        else
        {
            stat_layout = csv_src_lbtrm_stat_layout[modver].layout;
            stat_count = csv_src_lbtrm_stat_layout[modver].count;
        }
        memset((void *) &(Statistics->transport.lbtrm), 0, sizeof(lbm_src_transport_stats_t));
        for (idx = 0; idx < stat_count; ++idx)
        {
            ptr = next_csv_value(ptr, value, sizeof(value), fmt->mSeparator);
            if (ptr == NULL)
            {
                strncpy(ErrorString, "Data contains too few fields", sizeof(ErrorString));
                return (-1);
            }
            *((lbum_ulong_t *)(((unsigned char *)&(Statistics->transport.lbtrm)) + stat_layout + idx));
        }
        break;
    }

    case LBM_TRANSPORT_STAT_LBTRU:
        if (modver >= LBMMON_FORMAT_CSV_VERSION_CURRENT)
        {
            stat_layout = csv_src_lbtru_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].layout;
            stat_count = csv_src_lbtru_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].count;
        }
        else
        {
            stat_layout = csv_src_lbtru_stat_layout[modver].layout;
            stat_count = csv_src_lbtru_stat_layout[modver].count;
        }

```

```

        }
        memset((void *) &(Statistics->transport.lbtru), 0, sizeof(lbm_src_transport));
        for (idx = 0; idx < stat_count; ++idx)
        {
            ptr = next_csv_value(ptr, value, sizeof(value), fmt->mSeparator);
            if (ptr == NULL)
            {
                strncpy(ErrorString, "Data contains too few fields", sizeof(ErrorString));
                return (-1);
            }
            *((lbum_ulong_t *) (((unsigned char *) &(Statistics->transport.lbtru)) + idx)) = value;
        }
        break;

    case LBM_TRANSPORT_STAT_LBTIPC:
        if (modver >= LBMMON_FORMAT_CSV_VERSION_CURRENT)
        {
            stat_layout = csv_src_lbtipc_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT];
            stat_count = csv_src_lbtipc_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].count;
        }
        else
        {
            stat_layout = csv_src_lbtipc_stat_layout[modver].layout;
            stat_count = csv_src_lbtipc_stat_layout[modver].count;
        }
        memset((void *) &(Statistics->transport.lbtipc), 0, sizeof(lbm_src_transport));
        for (idx = 0; idx < stat_count; ++idx)
        {
            ptr = next_csv_value(ptr, value, sizeof(value), fmt->mSeparator);
            if (ptr == NULL)
            {
                strncpy(ErrorString, "Data contains too few fields", sizeof(ErrorString));
                return (-1);
            }
            *((lbum_ulong_t *) (((unsigned char *) &(Statistics->transport.lbtipc)) + idx)) = value;
        }
        break;

    case LBM_TRANSPORT_STAT_LBTRDMA:
        if (modver >= LBMMON_FORMAT_CSV_VERSION_CURRENT)
        {
            stat_layout = csv_src_lbtrdma_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT];
            stat_count = csv_src_lbtrdma_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].count;
        }
        else
        {
            stat_layout = csv_src_lbtrdma_stat_layout[modver].layout;
            stat_count = csv_src_lbtrdma_stat_layout[modver].count;
        }
        memset((void *) &(Statistics->transport.lbtrdma), 0, sizeof(lbm_src_transport));
        for (idx = 0; idx < stat_count; ++idx)
        {
            ptr = next_csv_value(ptr, value, sizeof(value), fmt->mSeparator);
            if (ptr == NULL)
            {
                strncpy(ErrorString, "Data contains too few fields", sizeof(ErrorString));
                return (-1);
            }
        }
    }
}

```

```

        }
        *((lbm_ulong_t *)(((unsigned char *)&(Statistics->transport.lbtrdma)) + st
    }
    break;

    default:
        strncpy(ErrorString, "Invalid LBM transport type", sizeof(ErrorString));
        return (-1);
    }
    return (0);
}

static size_t csv_evq_stat_offset_v2[] =
{
    offsetof(lbm_event_queue_stats_t, data_msgs),
    offsetof(lbm_event_queue_stats_t, data_msgs_tot),
    offsetof(lbm_event_queue_stats_t, data_msgs_svc_min),
    offsetof(lbm_event_queue_stats_t, data_msgs_svc_mean),
    offsetof(lbm_event_queue_stats_t, data_msgs_svc_max),
    offsetof(lbm_event_queue_stats_t, resp_msgs),
    offsetof(lbm_event_queue_stats_t, resp_msgs_tot),
    offsetof(lbm_event_queue_stats_t, resp_msgs_svc_min),
    offsetof(lbm_event_queue_stats_t, resp_msgs_svc_mean),
    offsetof(lbm_event_queue_stats_t, resp_msgs_svc_max),
    offsetof(lbm_event_queue_stats_t, topicless_im_msgs),
    offsetof(lbm_event_queue_stats_t, topicless_im_msgs_tot),
    offsetof(lbm_event_queue_stats_t, topicless_im_msgs_svc_min),
    offsetof(lbm_event_queue_stats_t, topicless_im_msgs_svc_mean),
    offsetof(lbm_event_queue_stats_t, topicless_im_msgs_svc_max),
    offsetof(lbm_event_queue_stats_t, wrcv_msgs),
    offsetof(lbm_event_queue_stats_t, wrcv_msgs_tot),
    offsetof(lbm_event_queue_stats_t, wrcv_msgs_svc_min),
    offsetof(lbm_event_queue_stats_t, wrcv_msgs_svc_mean),
    offsetof(lbm_event_queue_stats_t, wrcv_msgs_svc_max),
    offsetof(lbm_event_queue_stats_t, io_events),
    offsetof(lbm_event_queue_stats_t, io_events_tot),
    offsetof(lbm_event_queue_stats_t, io_events_svc_min),
    offsetof(lbm_event_queue_stats_t, io_events_svc_mean),
    offsetof(lbm_event_queue_stats_t, io_events_svc_max),
    offsetof(lbm_event_queue_stats_t, timer_events),
    offsetof(lbm_event_queue_stats_t, timer_events_tot),
    offsetof(lbm_event_queue_stats_t, timer_events_svc_min),
    offsetof(lbm_event_queue_stats_t, timer_events_svc_mean),
    offsetof(lbm_event_queue_stats_t, timer_events_svc_max),
    offsetof(lbm_event_queue_stats_t, source_events),
    offsetof(lbm_event_queue_stats_t, source_events_tot),
    offsetof(lbm_event_queue_stats_t, source_events_svc_min),
    offsetof(lbm_event_queue_stats_t, source_events_svc_mean),
    offsetof(lbm_event_queue_stats_t, source_events_svc_max),
    offsetof(lbm_event_queue_stats_t, unblock_events),
    offsetof(lbm_event_queue_stats_t, unblock_events_tot),
    offsetof(lbm_event_queue_stats_t, cancel_events),
    offsetof(lbm_event_queue_stats_t, cancel_events_tot),
    offsetof(lbm_event_queue_stats_t, cancel_events_svc_min),
    offsetof(lbm_event_queue_stats_t, cancel_events_svc_mean),
    offsetof(lbm_event_queue_stats_t, cancel_events_svc_max),
    offsetof(lbm_event_queue_stats_t, context_source_events),
}

```

```

        offsetof(lbm_event_queue_stats_t, context_source_events_tot),
        offsetof(lbm_event_queue_stats_t, context_source_events_svc_min),
        offsetof(lbm_event_queue_stats_t, context_source_events_svc_mean),
        offsetof(lbm_event_queue_stats_t, context_source_events_svc_max),
        offsetof(lbm_event_queue_stats_t, events),
        offsetof(lbm_event_queue_stats_t, events_tot),
        offsetof(lbm_event_queue_stats_t, age_min),
        offsetof(lbm_event_queue_stats_t, age_mean),
        offsetof(lbm_event_queue_stats_t, age_max)
    };
static size_t csv_evq_stat_offset_v3[] =
{
    offsetof(lbm_event_queue_stats_t, data_msgs),
    offsetof(lbm_event_queue_stats_t, data_msgs_tot),
    offsetof(lbm_event_queue_stats_t, data_msgs_svc_min),
    offsetof(lbm_event_queue_stats_t, data_msgs_svc_mean),
    offsetof(lbm_event_queue_stats_t, data_msgs_svc_max),
    offsetof(lbm_event_queue_stats_t, resp_msgs),
    offsetof(lbm_event_queue_stats_t, resp_msgs_tot),
    offsetof(lbm_event_queue_stats_t, resp_msgs_svc_min),
    offsetof(lbm_event_queue_stats_t, resp_msgs_svc_mean),
    offsetof(lbm_event_queue_stats_t, resp_msgs_svc_max),
    offsetof(lbm_event_queue_stats_t, topicless_im_msgs),
    offsetof(lbm_event_queue_stats_t, topicless_im_msgs_tot),
    offsetof(lbm_event_queue_stats_t, topicless_im_msgs_svc_min),
    offsetof(lbm_event_queue_stats_t, topicless_im_msgs_svc_mean),
    offsetof(lbm_event_queue_stats_t, topicless_im_msgs_svc_max),
    offsetof(lbm_event_queue_stats_t, wrcv_msgs),
    offsetof(lbm_event_queue_stats_t, wrcv_msgs_tot),
    offsetof(lbm_event_queue_stats_t, wrcv_msgs_svc_min),
    offsetof(lbm_event_queue_stats_t, wrcv_msgs_svc_mean),
    offsetof(lbm_event_queue_stats_t, wrcv_msgs_svc_max),
    offsetof(lbm_event_queue_stats_t, io_events),
    offsetof(lbm_event_queue_stats_t, io_events_tot),
    offsetof(lbm_event_queue_stats_t, io_events_svc_min),
    offsetof(lbm_event_queue_stats_t, io_events_svc_mean),
    offsetof(lbm_event_queue_stats_t, io_events_svc_max),
    offsetof(lbm_event_queue_stats_t, timer_events),
    offsetof(lbm_event_queue_stats_t, timer_events_tot),
    offsetof(lbm_event_queue_stats_t, timer_events_svc_min),
    offsetof(lbm_event_queue_stats_t, timer_events_svc_mean),
    offsetof(lbm_event_queue_stats_t, timer_events_svc_max),
    offsetof(lbm_event_queue_stats_t, source_events),
    offsetof(lbm_event_queue_stats_t, source_events_tot),
    offsetof(lbm_event_queue_stats_t, source_events_svc_min),
    offsetof(lbm_event_queue_stats_t, source_events_svc_mean),
    offsetof(lbm_event_queue_stats_t, source_events_svc_max),
    offsetof(lbm_event_queue_stats_t, unblock_events),
    offsetof(lbm_event_queue_stats_t, unblock_events_tot),
    offsetof(lbm_event_queue_stats_t, cancel_events),
    offsetof(lbm_event_queue_stats_t, cancel_events_tot),
    offsetof(lbm_event_queue_stats_t, cancel_events_svc_min),
    offsetof(lbm_event_queue_stats_t, cancel_events_svc_mean),
    offsetof(lbm_event_queue_stats_t, cancel_events_svc_max),
    offsetof(lbm_event_queue_stats_t, context_source_events),
    offsetof(lbm_event_queue_stats_t, context_source_events_tot),
    offsetof(lbm_event_queue_stats_t, context_source_events_svc_min),

```

```

        offsetof(lbm_event_queue_stats_t, context_source_events_svc_mean),
        offsetof(lbm_event_queue_stats_t, context_source_events_svc_max),
        offsetof(lbm_event_queue_stats_t, events),
        offsetof(lbm_event_queue_stats_t, events_tot),
        offsetof(lbm_event_queue_stats_t, age_min),
        offsetof(lbm_event_queue_stats_t, age_mean),
        offsetof(lbm_event_queue_stats_t, age_max),
        offsetof(lbm_event_queue_stats_t, callback_events),
        offsetof(lbm_event_queue_stats_t, callback_events_tot),
        offsetof(lbm_event_queue_stats_t, callback_events_svc_min),
        offsetof(lbm_event_queue_stats_t, callback_events_svc_mean),
        offsetof(lbm_event_queue_stats_t, callback_events_svc_max)
    };
#define csv_evq_stat_offset_v4 csv_evq_stat_offset_v3
static const lbmmon_csv_layout_t csv_evq_stat_layout [LBMMON_FORMAT_CSV_VERSION_CURRENT+1] =
{
    { NULL, 0 },
    { NULL, 0 },
    { csv_evq_stat_offset_v2, sizeof(csv_evq_stat_offset_v2)/sizeof(csv_evq_stat_offset_v2[0]) },
    { csv_evq_stat_offset_v3, sizeof(csv_evq_stat_offset_v3)/sizeof(csv_evq_stat_offset_v3[0]) },
    { csv_evq_stat_offset_v4, sizeof(csv_evq_stat_offset_v4)/sizeof(csv_evq_stat_offset_v4[0]) }
};

int
lbmon_evq_format_csv_deserialize(lbm_event_queue_stats_t * Statistics,
                                  const char * Source,
                                  size_t Length,
                                  unsigned short ModuleID,
                                  void * FormatClientData)
{
    const char * ptr;
    char value[1024];
    lbmmon_format_csv_t * fmt;
    size_t idx;
    unsigned char modid;
    unsigned char modver;
    const size_t * stat_layout = NULL;
    size_t stat_count = 0;

    if ((Statistics == NULL) || (Source == NULL) || (*Source == '\0') || (Length == 0) || (FormatClientData == NULL))
    {
        strncpy(ErrorString, "Invalid parameter", sizeof(ErrorString));
        return (-1);
    }
    fmt = (lbmmon_format_csv_t *) FormatClientData;

    modid = MODULE_ID(ModuleID);
    modver = MODULE_VERSION(ModuleID);
    if (modid != LBMMON_FORMAT_CSV_MODULE_ID)
    {
        strncpy(ErrorString, "Invalid module ID", sizeof(ErrorString));
        return (-1);
    }

    if (fmt->mBuffer == NULL)
    {
        fmt->mBufferSize = 1024;
    }
}

```

```

        fmt->mBuffer = malloc(fmt->mBufferSize);
    }
    if (Length >= fmt->mBufferSize)
    {
        fmt->mBufferSize = 2 * Length;
        free(fmt->mBuffer);
        fmt->mBuffer = malloc(fmt->mBufferSize);
    }
    memset(fmt->mBuffer, 0, fmt->mBufferSize);
    memcpy(fmt->mBuffer, Source, Length);
    ptr = fmt->mBuffer;
    if (modver >= LBMMON_FORMAT_CSV_VERSION_CURRENT)
    {
        stat_layout = csv_evq_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].layout;
        stat_count = csv_evq_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].count;
    }
    else
    {
        stat_layout = csv_evq_stat_layout[modver].layout;
        stat_count = csv_evq_stat_layout[modver].count;
    }
    memset((void *) Statistics, 0, sizeof(lbm_event_queue_stats_t));
    for (idx = 0; idx < stat_count; ++idx)
    {
        ptr = next_csv_value(ptr, value, sizeof(value), fmt->mSeparator);
        if (ptr == NULL)
        {
            strncpy(ErrorString, "Data contains too few fields", sizeof(ErrorString));
            return (-1);
        }
        *((lbm_ulong_t *) (((unsigned char *) Statistics) + stat_layout[idx])) = convert(
        )
        return (0);
    }

    static size_t csv_ctx_stat_offset_v2[] =
    {
        offsetof(lbm_context_stats_t, tr_dgrams_sent),
        offsetof(lbm_context_stats_t, tr_bytes_sent),
        offsetof(lbm_context_stats_t, tr_dgrams_rcved),
        offsetof(lbm_context_stats_t, tr_bytes_rcved),
        offsetof(lbm_context_stats_t, tr_dgrams_dropped_ver),
        offsetof(lbm_context_stats_t, tr_dgrams_dropped_type),
        offsetof(lbm_context_stats_t, tr_dgrams_dropped_malformed),
        offsetof(lbm_context_stats_t, tr_dgrams_send_failed),
        offsetof(lbm_context_stats_t, tr_src_topics),
        offsetof(lbm_context_stats_t, tr_rcv_topics),
        offsetof(lbm_context_stats_t, tr_rcv_unresolved_topics),
        offsetof(lbm_context_stats_t, lbtrm_unknown_msgs_rcved),
        offsetof(lbm_context_stats_t, lbtru_unknown_msgs_rcved),
        offsetof(lbm_context_stats_t, send_blocked),
        offsetof(lbm_context_stats_t, send_would_block),
        offsetof(lbm_context_stats_t, resp_blocked),
        offsetof(lbm_context_stats_t, resp_would_block)
    };
#define csv_ctx_stat_offset_v3 csv_ctx_stat_offset_v2
static size_t csv_ctx_stat_offset_v4[] =

```

```

{
    offsetof(lbm_context_stats_t, tr_dgrams_sent),
    offsetof(lbm_context_stats_t, tr_bytes_sent),
    offsetof(lbm_context_stats_t, tr_dgrams_rcved),
    offsetof(lbm_context_stats_t, tr_bytes_rcved),
    offsetof(lbm_context_stats_t, tr_dgrams_dropped_ver),
    offsetof(lbm_context_stats_t, tr_dgrams_dropped_type),
    offsetof(lbm_context_stats_t, tr_dgrams_dropped_malformed),
    offsetof(lbm_context_stats_t, tr_dgrams_send_failed),
    offsetof(lbm_context_stats_t, tr_src_topics),
    offsetof(lbm_context_stats_t, tr_rcv_topics),
    offsetof(lbm_context_stats_t, tr_rcv_unresolved_topics),
    offsetof(lbm_context_stats_t, lbtrm_unknown_msgs_rcved),
    offsetof(lbm_context_stats_t, lbtru_unknown_msgs_rcved),
    offsetof(lbm_context_stats_t, send_blocked),
    offsetof(lbm_context_stats_t, send_would_block),
    offsetof(lbm_context_stats_t, resp_blocked),
    offsetof(lbm_context_stats_t, resp_would_block),
    offsetof(lbm_context_stats_t, uim_dup_msgs_rcved),
    offsetof(lbm_context_stats_t, uim_msgs_no_stream_rcved)
};

static const lbmon_csv_layout_t csv_ctx_stat_layout[LBMON_FORMAT_CSV_VERSION_CURRENT+1] =
{
    { NULL, 0 },
    { NULL, 0 },
    { csv_ctx_stat_offset_v2, sizeof(csv_ctx_stat_offset_v2)/sizeof(csv_ctx_stat_offset_v2[0]) },
    { csv_ctx_stat_offset_v3, sizeof(csv_ctx_stat_offset_v3)/sizeof(csv_ctx_stat_offset_v3[0]) },
    { csv_ctx_stat_offset_v4, sizeof(csv_ctx_stat_offset_v4)/sizeof(csv_ctx_stat_offset_v4[0]) }
};

int
lbmon_ctx_format_csv_deserialize(lbm_context_stats_t * Statistics,
                                  const char * Source,
                                  size_t Length,
                                  unsigned short ModuleID,
                                  void * FormatClientData)
{
    const char * ptr;
    char value[1024];
    lbmon_format_csv_t * fmt;
    size_t idx;
    unsigned char modid;
    unsigned char modver;
    const size_t * stat_layout = NULL;
    size_t stat_count = 0;

    if ((Statistics == NULL) || (Source == NULL) || (*Source == '\0') || (Length == 0) || (FormatClientData == NULL))
    {
        strncpy(ErrorString, "Invalid parameter", sizeof(ErrorString));
        return (-1);
    }
    fmt = (lbmon_format_csv_t *) FormatClientData;

    modid = MODULE_ID(ModuleID);
    modver = MODULE_VERSION(ModuleID);
    if (modid != LBMON_FORMAT_CSV_MODULE_ID)
    {
}

```

```

        strncpy(ErrorString, "Invalid module ID", sizeof(ErrorString));
        return (-1);
    }

    if (fmt->mBuffer == NULL)
    {
        fmt->mBufferSize = 1024;
        fmt->mBuffer = malloc(fmt->mBufferSize);
    }
    if (Length >= fmt->mBufferSize)
    {
        fmt->mBufferSize = 2 * Length;
        free(fmt->mBuffer);
        fmt->mBuffer = malloc(fmt->mBufferSize);
    }
    memset(fmt->mBuffer, 0, fmt->mBufferSize);
    memcpy(fmt->mBuffer, Source, Length);
    ptr = fmt->mBuffer;
    if (modver >= LBMMON_FORMAT_CSV_VERSION_CURRENT)
    {
        stat_layout = csv_ctx_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].layout;
        stat_count = csv_ctx_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].count;
    }
    else
    {
        stat_layout = csv_ctx_stat_layout[modver].layout;
        stat_count = csv_ctx_stat_layout[modver].count;
    }
    memset((void *) Statistics, 0, sizeof(lbm_context_stats_t));
    for (idx = 0; idx < stat_count; ++idx)
    {
        ptr = next_csv_value(ptr, value, sizeof(value), fmt->mSeparator);
        if (ptr == NULL)
        {
            strncpy(ErrorString, "Data contains too few fields", sizeof(ErrorString));
            return (-1);
        }
        *((lbm_ulong_t *) (((unsigned char *) Statistics) + stat_layout[idx])) = convert(
    }
    return (0);
}

int
lbmmmon_format_csv_finish(void * FormatClientData)
{
    if (FormatClientData != NULL)
    {
        lbmmmon_format_csv_t * data = (lbmmmon_format_csv_t *) FormatClientData;
        if (data->mBuffer != NULL)
        {
            free(data->mBuffer);
            data->mBuffer = NULL;
        }
        free(data);
    }
    return (0);
}

```

```
const char *
lbmon_format_csv_errmsg(void)
{
    return (ErrorString);
}
```

## 9.11 LBMMON LBMSNMP transport module

- [lbmmmonrlbmsnmp.h](#)
- [lbmmmonrlbmsnmp.c](#)

## 9.12 Source code for lbmmmontrlbmsnmp.h

```
/** \file lbmmmontrlbmsnmp.h
   \brief Ultra Messaging (UM) Monitoring API
   \author David K. Ameiss - Informatica Corporation
   \version $Id: //UMprod/REL_5_3_6/29West/lbm/src/mon/lbm/lbmmmontrlbmsnmp.h#2 $

The Ultra Messaging (UM) Monitoring API Description. Included
are types, constants, and functions related to the API. Contents are
subject to change.

All of the documentation and software included in this and any
other Informatica Corporation Ultra Messaging Releases
Copyright (C) Informatica Corporation. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted only as covered by the terms of a
valid software license agreement with Informatica Corporation.

Copyright (C) 2006-2014, Informatica Corporation. All Rights Reserved.

THE SOFTWARE IS PROVIDED "AS IS" AND INFORMATICA DISCLAIMS ALL WARRANTIES
EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF
NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR
PURPOSE. INFORMATICA DOES NOT WARRANT THAT USE OF THE SOFTWARE WILL BE
UNINTERRUPTED OR ERROR-FREE. INFORMATICA SHALL NOT, UNDER ANY CIRCUMSTANCES, BE
LIABLE TO LICENSEE FOR LOST PROFITS, CONSEQUENTIAL, INCIDENTAL, SPECIAL OR
INDIRECT DAMAGES ARISING OUT OF OR RELATED TO THIS AGREEMENT OR THE
TRANSACTIONS CONTEMPLATED HEREUNDER, EVEN IF INFORMATICA HAS BEEN APPRISED OF
THE LIKELIHOOD OF SUCH DAMAGES.

*/
#ifndef LBMMONTRLBMSNMP_H
#define LBMMONTRLBMSNMP_H

#include <stdlib.h>
#ifdef _WIN32
    #include <winsock2.h>
#endif
#include <lbm/lbmmmon.h>

#if defined(__cplusplus)
extern "C" {
#endif /* __cplusplus */

/*! \brief Return a pointer to the LBMMON_TRANSPORT_LBMSNMP module structure.

\return Pointer to LBMMON_TRANSPORT_LBMSNMP.

*/
LBMExpDLL const lbmmmon_transport_func_t * lbmmmon_transport_lbmsnmp_module(void);

/*! \brief Initialize the LBM SNMP transport module to send statistics.

\param TransportClientData A pointer which may be filled in (by this function) with
a pointer to transport-specific client data.
\param TransportOptions The TransportOptions argument originally passed to
lbmmmon_sctl_create().

```

```

        \return Zero if successful, -1 otherwise.
*/
LBMEExpDLL int lbmmon_transport_lbmsnmp_initsrc(void * * TransportClientData,
    cons

/*! \brief Initialize the LBM SNMP transport module to receive statistics.

\param TransportClientData A pointer which may be filled in (by this function) with
    a pointer to transport-specific client data.
\param TransportOptions The TransportOptions argument originally passed to
    lbmmon_sctl_create().
\return Zero if successful, -1 otherwise.
*/
LBMEExpDLL int lbmmon_transport_lbmsnmp_initrcv(void * * TransportClientData,
    cons

/*! \brief Send a statistics packet.

\param Data The data to be sent.
\param Length The length of the data.
\param TransportClientData A pointer to transport-specific client data.
\return Zero if successful, -1 otherwise.
*/
LBMEExpDLL int lbmmon_transport_lbmsnmp_send(const char * Data,
    size_t
    void *

/*! \brief Receive statistics packet data.

\param Data A pointer to a buffer to receive the packet data.
\param Length A pointer to a size_t. On entry, it contains the maximum number of bytes
    to receive. On exit, it contains the actual number of bytes received.
\param TimeoutMS Maximum timeout in milliseconds. If no data is available within
    the timeout value, return.
\param TransportClientData A pointer to transport-specific client data.
\return Zero if successful, -1 otherwise.
*/
LBMEExpDLL int lbmmon_transport_lbmsnmp_receive(char * Data,
    size_t
    unsi
    voi

/*! \brief Finish LBM SNMP transport module source processing.

\param TransportClientData A pointer to transport-specific client data.
\return Zero if successful, -1 otherwise.
*/
LBMEExpDLL int lbmmon_transport_lbmsnmp_src_finish(void * TransportClientData);

/*! \brief Finish LBM SNMP transport module receiver processing.

\param TransportClientData A pointer to transport-specific client data.
\return Zero if successful, -1 otherwise.
*/
LBMEExpDLL int lbmmon_transport_lbmsnmp_rcv_finish(void * TransportClientData);

/*! \brief Return a messages describing the last error encountered.

```

```
\return A string containing a description of the last error encountered by the module.  
*/  
LBMExpDLL const char * lbmon_transport_lbmsnmp_errmsg(void);  
  
#if defined(__cplusplus)  
}  
#endif /* __cplusplus */  
  
#endif
```

## 9.13 Source code for lbmmmonrlbmsnmp.c

```
/*
All of the documentation and software included in this and any
other Informatica Corporation Ultra Messaging Releases
Copyright (C) Informatica Corporation. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted only as covered by the terms of a
valid software license agreement with Informatica Corporation.

Copyright (C) 2004-2014, Informatica Corporation. All Rights Reserved.

THE SOFTWARE IS PROVIDED "AS IS" AND INFORMATICA DISCLAIMS ALL WARRANTIES
EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF
NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR
PURPOSE. INFORMATICA DOES NOT WARRANT THAT USE OF THE SOFTWARE WILL BE
UNINTERRUPTED OR ERROR-FREE. INFORMATICA SHALL NOT, UNDER ANY CIRCUMSTANCES, BE
LIABLE TO LICENSEE FOR LOST PROFITS, CONSEQUENTIAL, INCIDENTAL, SPECIAL OR
INDIRECT DAMAGES ARISING OUT OF OR RELATED TO THIS AGREEMENT OR THE
TRANSACTIONS CONTEMPLATED HEREUNDER, EVEN IF INFORMATICA HAS BEEN APPRISED OF
THE LIKELIHOOD OF SUCH DAMAGES.

*/
#ifndef __VOS__
#define _POSIX_C_SOURCE 200112L
#include <sys/time.h>
#endif

#include <stdio.h>
#include <time.h>
#include <string.h>
#ifdef _WIN32
#define strcasecmp strcmp
#define sprintf _snprintf
#else
#include "config.h"
#include <unistd.h>
#if defined(__TANDEM)
#if defined(HAVE_TANDEM_SPT)
#include <ktdmtyp.h>
#include <spthread.h>
#else
#include <pthread.h>
#endif
#else
#include <pthread.h>
#endif
#endif
#include <strings.h>
#endif
#include <lbm/lbmmmon.h>
#include <lbm/lbmmmonrlbmsnmp.h>
#include <lbm/lbmaux.h>

/*
Package all of the needed function pointers for this module into a
```

```

        lbmon_transport_func_t structure.

*/
static const lbmon_transport_func_t LBMMON_TRANSPORT_LBMSNMP =
{
    lbmon_transport_lbmsnmp_initsrc,
    lbmon_transport_lbmsnmp_initrcv,
    lbmon_transport_lbmsnmp_send,
    lbmon_transport_lbmsnmp_receive,
    lbmon_transport_lbmsnmp_src_finish,
    lbmon_transport_lbmsnmp_rcv_finish,
    lbmon_transport_lbmsnmp_errmsg
};

/*
     For a statistics source, one of these gets returned as the TransportClientData.
*/
typedef struct
{
    /* LBM context attributes */
    lbm_context_attr_t * mContextAttributes;
    /* LBM context created to send a statistics packet */
    lbm_context_t * mContext;
    /* LBM topic attributes */
    lbm_src_topic_attr_t * mTopicAttributes;
    /* LBM source created to send a statistics packet */
    lbm_src_t * mSource;
    /* LBM topic */
    lbm_topic_t * mTopic;
} lbmon_transport_lbmsnmp_src_t;

/*
     A queue of incoming statistics packets is maintained. This describes each
     entry in the queue.
*/
struct lbmon_transport_lbmsnmp_rcv_node_t_stct
{
    /* Pointer to the LBM message */
    lbm_msg_t * mMessage;
    /* Number of bytes of the message returned to caller */
    size_t mUsedBytes;
    /* Next entry in the queue */
    struct lbmon_transport_lbmsnmp_rcv_node_t_stct * mNext;
};
typedef struct lbmon_transport_lbmsnmp_rcv_node_t_stct lbmon_transport_lbmsnmp_rcv_node_t;

/*
     For a statistics receiver, one of these gets returned as the TransportClientData.
*/
typedef struct
{
    /* Flag to indicate lock has been created */
    unsigned int mLockCreated;
    /* Lock to prevent access by multiple threads */
#ifdef _WIN32
    CRITICAL_SECTION mLock;
#else
    pthread_mutex_t mLock;

```

```

#endif

/* LBM context attributes */
lbm_context_attr_t * mContextAttributes;
/* LBM context used to receive packets */
lbm_context_t * mContext;
/* LBM receiver used to receive packets */
lbm_rcv_t * mReceiver;
/* Topic attributes */
lbm_rcv_topic_attr_t * mTopicAttributes;
/* Topic */
lbm_topic_t * mTopic;
/* Wildcard receiver attributes */
lbm_wildcard_rcv_attr_t * mWildcardReceiverAttributes;
/* If we're using a wildcard receiver... */
lbm_wildcard_rcv_t * mWildcardReceiver;
/* Head of the message queue */
lbmmmon_transport_lbmsnmp_rcv_node_t * mHead;
/* Tail of the message queue */
lbmmmon_transport_lbmsnmp_rcv_node_t * mTail;
} lbmmmon_transport_lbmsnmp_rcv_t;

static void      src_cleanup(lbmmmon_transport_lbmsnmp_src_t * Data);
static void      rcv_cleanup(lbmmmon_transport_lbmsnmp_rcv_t * Data);
static int receive_callback(lbm_rcv_t * Receiver, lbm_msg_t * Message, void * ClientData);
static void lock_receiver(lbmmmon_transport_lbmsnmp_rcv_t * Receiver);
static void unlock_receiver(lbmmmon_transport_lbmsnmp_rcv_t * Receiver);
static int scope_is_valid(const char * Scope);

#define DEFAULT_CONTEXT_NAME "29west_statistics_context"
#define DEFAULT_TOPIC "/29west/statistics"
#define DEFAULT_MULTICAST_TTL "0"
#define DEFAULT_TOPIC_RESOLUTION_ADDRESS "225.200.200.200"
#define DEFAULT_LBTMRM_ADDRESS "225.200.200.201"

static char ErrorString[1024];

typedef struct
{
    const char * option;
    const char * value;
} option_entry_t;

static option_entry_t SourceContextOption[] =
{
    /* Force embedded mode for simplicity. */
    { "operational_mode", "embedded" },
    /* Disable monitoring for this context. */
    { "monitor_interval", "0" },
    /* We don't need request/response, so don't use up ports. */
    { "request_tcp_bind_request_port", "0" },
    /* We don't need MIM, so disable MIM receiver. */
    { "mim_incoming_address", "0.0.0.0" },
    /* No need to cache topics. */
    { "resolver_cache", "0" },
    /* Force TTL=0 to keep stats and advertisements on the local machine. */
    { "resolver_multicast_ttl", DEFAULT_MULTICAST_TTL },
    /* Use a specific topic resolution address. */
}

```

```
{ "resolver_multicast_address", DEFAULT_TOPIC_RESOLUTION_ADDRESS },
/* End of list. */
{ NULL, NULL }
};

static option_entry_t SourceContextOptionFixed[] =
{
    /* Force embedded mode for simplicity. */
    { "operational_mode", "embedded" },
    /* Disable monitoring for this context. */
    { "monitor_interval", "0" },
    /* We don't need request/response, so don't use up ports. */
    { "request_tcp_bind_request_port", "0" },
    /* We don't need MIM, so disable MIM receiver. */
    { "mim_incoming_address", "0.0.0.0" },
    /* No need to cache topics. */
    { "resolver_cache", "0" },
    /* End of list. */
    { NULL, NULL }
};

static option_entry_t ReceiverContextOption[] =
{
    /* Force embedded mode for simplicity. */
    { "operational_mode", "embedded" },
    /* Disable monitoring for this context. */
    { "monitor_interval", "0" },
    /* We don't need request/response, so don't use up ports. */
    { "request_tcp_bind_request_port", "0" },
    /* We don't need MIM, so disable MIM receiver. */
    { "mim_incoming_address", "0.0.0.0" },
    /* No need to cache topics. */
    { "resolver_cache", "0" },
    /* Force TTL=0. */
    { "resolver_multicast_ttl", DEFAULT_MULTICAST_TTL },
    /* Use a specific topic resolution address. */
    { "resolver_multicast_address", DEFAULT_TOPIC_RESOLUTION_ADDRESS },
    /* End of list. */
    { NULL, NULL }
};

static option_entry_t ReceiverContextOptionFixed[] =
{
    /* Force embedded mode for simplicity. */
    { "operational_mode", "embedded" },
    /* Disable monitoring for this context. */
    { "monitor_interval", "0" },
    /* We don't need request/response, so don't use up ports. */
    { "request_tcp_bind_request_port", "0" },
    /* We don't need MIM, so disable MIM receiver. */
    { "mim_incoming_address", "0.0.0.0" },
    /* No need to cache topics. */
    { "resolver_cache", "0" },
    /* End of list. */
    { NULL, NULL }
};
```

```

static option_entry_t SourceTopicOption[] =
{
    /* Minimize memory used for LBT-RU retransmissions. */
    { "transport_lbtru_transmission_window_size", "500000" },
    /* Minimize memory used for LBT-RM retransmissions. */
    { "transport_lbtrm_transmission_window_size", "500000" },
    /* Force LBT-RM. */
    { "transport", "lbtrm" },
    /* Force the LBT-RM address. */
    { "transport_lbtrm_multicast_address", DEFAULT_LBTRM_ADDRESS },
    /* End of list. */
    { NULL, NULL }
};

static option_entry_t ReceiverTopicOption[] =
{
    /* End of list. */
    { NULL, NULL }
};

static option_entry_t WildcardReceiverOption[] =
{
    /* End of list. */
    { NULL, NULL }
};

const lbmon_transport_func_t *
lbmon_transport_lbmsnmp_module(void)
{
    return (&LBMMON_TRANSPORT_LBMSNMP);
}

int
lbmon_transport_lbmsnmp_initsrc(void * * TransportClientData, const void * TransportOptions)
{
    lbmon_transport_lbmsnmp_src_t * data;
    int rc;
    const char * ptr = (const char *) TransportOptions;
    char key[512];
    char value[512];
    char config_file[512];
    char topic[512];
    char scope[512];
    char option[512];
    option_entry_t * entry;

    memset(ErrorString, 0, sizeof(ErrorString));
    data = malloc(sizeof(lbmon_transport_lbmsnmp_src_t));
    data->mContextAttributes = NULL;
    data->mContext = NULL;
    data->mTopicAttributes = NULL;
    data->mSource = NULL;
    data->mTopic = NULL;

    /* Process any options */
    memset(config_file, 0, sizeof(config_file));
    strncpy(topic, DEFAULT_TOPIC, sizeof(topic));
}

```

```

while ((ptr = lbmon_next_key_value_pair(ptr, key, sizeof(key), value, sizeof(value))) != NULL)
{
    if (strcasecmp(key, "config") == 0)
    {
        strncpy(config_file, value, sizeof(config_file));
    }
    else if (strcasecmp(key, "topic") == 0)
    {
        strncpy(topic, value, sizeof(topic));
    }
}

/* Initialize the context attributes */
rc = lbm_context_attr_create_default(&(data->mContextAttributes));
if (rc == LBM_FAILURE)
{
    snprintf(ErrorString,
             sizeof(ErrorString),
             "lbm_context_attr_init() failed, %s",
             lbm_errmsg());
    return (rc);
}
/* Set the default context name */
rc = lbm_context_attr_str_setopt(data->mContextAttributes, "context_name", DEFAULT_CONTEXT_NAME);
if (rc == LBM_FAILURE)
{
    snprintf(ErrorString,
             sizeof(ErrorString),
             "lbm_context_attr_str_setopt() failed, %s",
             lbm_errmsg());
    return (rc);
}
entry = &SourceContextOption[0];
while (entry->option != NULL)
{
    rc = lbm_context_attr_str_setopt(data->mContextAttributes, entry->option, entry->value);
    if (rc == LBM_FAILURE)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),
                 "error setting option [context %s %s], %s",
                 entry->option,
                 entry->value,
                 lbm_errmsg());
        src_cleanup(data);
        return (rc);
    }
    entry++;
}

/* Create the context */
if (config_file[0] != '\0')
{
    /* A config file was passed as an option. Use it to populate the context attributes. */
    rc = lbmaux_context_attr_setopt_from_file(data->mContextAttributes, config_file);
    if (rc != 0)
    {

```

```

        sprintf(ErrorString,
                sizeof(ErrorString),
                "lbmaux_context_attr_setopt_from_file() failed, %s",
                lbm_errmsg());
        src_cleanup(data);
        return (-1);
    }
}
/* Go back through the options, looking for any specific context options. */
ptr = (const char *) TransportOptions;
while ((ptr = lbmmmon_next_key_value_pair(ptr, key, sizeof(key), value, sizeof(value))) != NULL)
{
    if (sscanf(key, "%[a-zA-Z_]%[a-zA-Z_]", scope, option) != 2)
    {
        continue;
    }
    if (scope_is_valid(scope) == -1)
    {
        sprintf(ErrorString,
                sizeof(ErrorString),
                "invalid option scope [%s]", scope);
        src_cleanup(data);
        return (-1);
    }
    if (strcasecmp(scope, "context") == 0)
    {
        rc = lbm_context_attr_str_setopt(data->mContextAttributes, option, value);
        if (rc == LBM_FAILURE)
        {
            sprintf(ErrorString,
                    sizeof(ErrorString),
                    "invalid option [context %s %s], %s", option, value,
                    lbm_errmsg());
            src_cleanup(data);
            return (rc);
        }
    }
}

entry = &SourceContextOptionFixed[0];
while (entry->option != NULL)
{
    rc = lbm_context_attr_str_setopt(data->mContextAttributes, entry->option, entry->value);
    if (rc == LBM_FAILURE)
    {
        sprintf(ErrorString,
                sizeof(ErrorString),
                "error setting option [context %s %s], %s", entry->option, entry->value,
                lbm_errmsg());
        src_cleanup(data);
        return (rc);
    }
}

```

```

        entry++;
    }

/* Create the context */
rc = lbm_context_create(&(data->mContext), data->mContextAttributes, NULL, NULL);
if (rc == LBM_FAILURE)
{
    snprintf(ErrorString,
             sizeof(ErrorString),
             "lbm_context_create() failed, %s",
             lbm_errmsg());
    src_cleanup(data);
    return (rc);
}

/* Initialize the source topic attributes */
rc = lbm_src_topic_attr_create_default(&(data->mTopicAttributes));
if (rc == LBM_FAILURE)
{
    snprintf(ErrorString,
             sizeof(ErrorString),
             "lbm_src_topic_attr_create_default() failed, %s",
             lbm_errmsg());
    src_cleanup(data);
    return (rc);
}

/* Apply the default options first */
entry = &SourceTopicOption[0];
while (entry->option != NULL)
{
    rc = lbm_src_topic_attr_str_setopt(data->mTopicAttributes, entry->option, entry->value);
    if (rc == LBM_FAILURE)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),
                 "error setting option [source %s %s], %s",
                 entry->option,
                 entry->value,
                 lbm_errmsg());
        src_cleanup(data);
        return (rc);
    }
    entry++;
}
/* Overwrite the transport options if they are Part of config file */
if (config_file[0] != '\0')
{
    /* A config file was passed as an option. Use it to populate the source topic attributes.
    rc = lbmaux_src_topic_attr_setopt_from_file(data->mTopicAttributes, config_file);
    if (rc != 0)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),
                 "lbmaux_src_topic_attr_setopt_from_file() failed, %s",
                 lbm_errmsg());
        src_cleanup(data);
        return (-1);
    }
}

```

```

        }

    /* Go back through the options, looking for any specific source options. */
    ptr = (const char *) TransportOptions;
    while ((ptr = lbmmon_next_key_value_pair(ptr, key, sizeof(key), value, sizeof(value))) != NULL)
    {
        if (sscanf(key, "%[a-zA-Z_]|%[a-zA-Z_]", scope, option) != 2)
        {
            continue;
        }
        if (strcasecmp(scope, "source") == 0)
        {
            rc = lbm_src_topic_attr_str_setopt(data->mTopicAttributes, option, value);
            if (rc == LBM_FAILURE)
            {
                snprintf(ErrorString,
                         sizeof(ErrorString),
                         "invalid option [source %s %s], %s",
                         option,
                         value,
                         lbm_errmsg());
                src_cleanup(data);
                return (rc);
            }
        }
    }
    /* Create the topic */
    rc = lbm_src_topic_alloc(&(data->mTopic), data->mContext, topic, data->mTopicAttributes);
    if (rc == LBM_FAILURE)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),
                 "lbm_src_topic_alloc() failed, %s",
                 lbm_errmsg());
        src_cleanup(data);
        return (rc);
    }

    /* Create the source */
    rc = lbm_src_create(&(data->mSource), data->mContext, data->mTopic, NULL, NULL, NULL);
    if (rc == LBM_FAILURE)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),
                 "lbm_src_create() failed, %s",
                 lbm_errmsg());
        src_cleanup(data);
        return (rc);
    }

    /* Pass back the lbmmon_transport_lbmsnmp_src_t created */
    *TransportClientData = data;
    return (0);
}

/*
This function is called upon receipt of an LBM message (when operating as

```

```

        a statistics receiver).

*/
int
receive_callback(lbm_rcv_t * Receiver, lbm_msg_t * Message, void * ClientData)
{
    lbmmon_transport_lbmsnmp_rcv_t * rcv = (lbmmon_transport_lbmsnmp_rcv_t *) ClientData;
    lbmmon_transport_lbmsnmp_rcv_node_t * node;

    if (Message->type == LBM_MSG_DATA)
    {
        /* A data message. We want to enqueue it for processing. */
        lock_receiver(rcv);
        node = malloc(sizeof(lbmmon_transport_lbmsnmp_rcv_node_t));
        /*
           Since we hold onto the message until it is actually processed,
           let LBM know about it.
        */
        lbm_msg_retain(Message);
        node->mMessage = Message;
        node->mUsedBytes = 0; /* No data returned as yet */

        /* Link the message onto the queue */
        node->mNext = NULL;
        if (rcv->mTail != NULL)
        {
            rcv->mTail->mNext = node;
        }
        else
        {
            rcv->mHead = node;
        }
        rcv->mTail = node;
        unlock_receiver(rcv);
    }
    return (0);
}

int
lbmmon_transport_lbmsnmp_initrcv(void ** TransportClientData, const void * TransportOptions)
{
    lbmmon_transport_lbmsnmp_rcv_t * data;
    int rc;
    const char * ptr = (const char *) TransportOptions;
    char key[512];
    char value[512];
    char config_file[512];
    char topic[512];
    char wildcard_topic[512];
    char scope[512];
    char option[512];
    option_entry_t * entry;

    memset(ErrorString, 0, sizeof(ErrorString));
    data = malloc(sizeof(lbmmon_transport_lbmsnmp_rcv_t));

    data->mLockCreated = 0;
    data->mContextAttributes = NULL;
}

```

```

data->mContext = NULL;
data->mReceiver = NULL;
data->mTopicAttributes = NULL;
data->mTopic = NULL;
data->mWildcardReceiverAttributes = NULL;
data->mWildcardReceiver = NULL;
data->mHead = NULL;
data->mTail = NULL;

/* Process any options */
memset(config_file, 0, sizeof(config_file));
strncpy(topic, DEFAULT_TOPIC, sizeof(topic));
memset(wildcard_topic, 0, sizeof(wildcard_topic));
while ((ptr = lbmmmon_next_key_value_pair(ptr, key, sizeof(key), value, sizeof(value))) != NULL)
{
    if (strcasecmp(key, "config") == 0)
    {
        strncpy(config_file, value, sizeof(config_file));
    }
    else if (strcasecmp(key, "topic") == 0)
    {
        strncpy(topic, value, sizeof(topic));
    }
    else if (strcasecmp(key, "wctopic") == 0)
    {
        strncpy(wildcard_topic, value, sizeof(wildcard_topic));
    }
}

/* Initialize the context attributes */
rc = lbm_context_attr_create_default(&(data->mContextAttributes));
if (rc == LBM_FAILURE)
{
    sprintf(ErrorString,
            sizeof(ErrorString),
            "lbm_context_attr_init() failed, %s",
            lbm_errmsg());
    rcv_cleanup(data);
    return (rc);
}
/* Set the default context name */
rc = lbm_context_attr_str_setopt(data->mContextAttributes, "context_name", DEFAULT_CONTEXT_NAME);
if (rc == LBM_FAILURE)
{
    sprintf(ErrorString,
            sizeof(ErrorString),
            "lbm_context_attr_str_setopt() failed, %s",
            lbm_errmsg());
    return (rc);
}
/* Populate with Default Values */
entry = &ReceiverContextOption[0];
while (entry->option != NULL)
{
    rc = lbm_context_attr_str_setopt(data->mContextAttributes, entry->option, entry->value);
    if (rc == LBM_FAILURE)
    {

```

```

        snprintf(ErrorString,
                  sizeof(ErrorString),
                  "error setting option [context %s %s], %s",
                  entry->option,
                  entry->value,
                  lbm_errmsg());
        rcv_cleanup(data);
        return (rc);
    }
    entry++;
}

/* Create the context */
if (config_file[0] != '\0')
{
    /* A config file was passed as an option. Use it to populate the context attributes. */
    rc = lbmaux_context_attr_setopt_from_file(data->mContextAttributes, config_file);
    if (rc != 0)
    {
        snprintf(ErrorString,
                  sizeof(ErrorString),
                  "lbmaux_context_attr_setopt_from_file() failed, %s",
                  lbm_errmsg());
        rcv_cleanup(data);
        return (-1);
    }
}
/* Go back through the options, looking for any specific context options. */
ptr = (const char *) TransportOptions;
while ((ptr = lbmon_next_key_value_pair(ptr, key, sizeof(key), value, sizeof(value))) != NULL)
{
    if (sscanf(key, "%[a-zA-Z_]%[a-zA-Z_]", scope, option) != 2)
    {
        continue;
    }
    if (scope_is_valid(scope) == -1)
    {
        snprintf(ErrorString,
                  sizeof(ErrorString),
                  "invalid option scope [%s]",
                  scope);
        rcv_cleanup(data);
        return (-1);
    }
    if (strcasecmp(scope, "context") == 0)
    {
        rc = lbm_context_attr_str_setopt(data->mContextAttributes, option, value);
        if (rc == LBM_FAILURE)
        {
            snprintf(ErrorString,
                  sizeof(ErrorString),
                  "invalid option [context %s %s], %s",
                  option,
                  value,
                  lbm_errmsg());
            rcv_cleanup(data);
            return (rc);
        }
    }
}

```

```

        }
    }

entry = &ReceiverContextOptionFixed[0];
while (entry->option != NULL)
{
    rc = lbm_context_attr_str_setopt(data->mContextAttributes, entry->option, entry-
    if (rc == LBM_FAILURE)
    {
        sprintf(ErrorString,
                sizeof(ErrorString),
                "error setting option [context %s %s], %s",
                entry->option,
                entry->value,
                lbm_errmsg());
        rcv_cleanup(data);
        return (rc);
    }
    entry++;
}

/* Create the context */
rc = lbm_context_create(&(data->mContext), data->mContextAttributes, NULL, NULL);
if (rc == LBM_FAILURE)
{
    sprintf(ErrorString,
            sizeof(ErrorString),
            "lbm_context_create() failed, %s",
            lbm_errmsg());
    rcv_cleanup(data);
    return (rc);
}

/* If a wildcard topic was specified, initialize the wildcard receiver attributes. */
if (wildcard_topic[0] != '\0')
{
    rc = lbm_wildcard_rcv_attr_create_default(&(data->mWildcardReceiverAttributes));
    if (rc == LBM_FAILURE)
    {
        sprintf(ErrorString,
                sizeof(ErrorString),
                "lbm_wildcard_rcv_attr_init() failed, %s",
                lbm_errmsg());
        rcv_cleanup(data);
        return (rc);
    }
    if (config_file[0] != '\0')
    {
        /* A config file was passed as an option. Use it to populate the wildcard receiver attributes. */
        rc = lbmaux_wildcard_rcv_attr_setopt_from_file(data->mWildcardReceiverAttributes);
        if (rc == LBM_FAILURE)
        {
            sprintf(ErrorString,
                    sizeof(ErrorString),
                    "lbmaux_wildcard_rcv_attr_setopt_from_file() failed, %s",
                    lbm_errmsg());
            rcv_cleanup(data);
            return (rc);
        }
    }
}

```

```

        rcv_cleanup(data);
        return (-1);
    }
}
/* Go back through the options, looking for any specific wildcard receiver options. */
ptr = (const char *) TransportOptions;
while ((ptr = lbmmmon_next_key_value_pair(ptr, key, sizeof(key), value, sizeof(value))) !=
{
    if (sscanf(key, "%[a-zA-Z_]%[a-zA-Z_]", scope, option) != 2)
    {
        continue;
    }
    if (strcasecmp(scope, "wildcard_receiver") == 0)
    {
        rc = lbm_wildcard_rcv_attr_str_setopt(data->mWildcardReceiverAttributes, option,
        if (rc == LBM_FAILURE)
        {
            sprintf(ErrorString,
                    sizeof(ErrorString),
                    "invalid option [wildcard_receiver %s %s], %s",
                    option,
                    value,
                    lbm_errmsg());
            rcv_cleanup(data);
            return (rc);
        }
    }
    entry = &WildcardReceiverOption[0];
    while (entry->option != NULL)
    {
        rc = lbm_wildcard_rcv_attr_str_setopt(data->mWildcardReceiverAttributes, entry->option,
        if (rc == LBM_FAILURE)
        {
            sprintf(ErrorString,
                    sizeof(ErrorString),
                    "error setting option [wildcard_receiver %s %s], %s",
                    entry->option,
                    entry->value,
                    lbm_errmsg());
            rcv_cleanup(data);
            return (rc);
        }
    }
    entry++;
}

/* Initialize and set the receiver topic attributes. */
rc = lbm_rcv_topic_attr_create_default(&(data->mTopicAttributes));
if (rc == LBM_FAILURE)
{
    sprintf(ErrorString,
            sizeof(ErrorString),
            "lbm_rcv_topic_attr_init() failed, %s",
            lbm_errmsg());
    rcv_cleanup(data);
    return (rc);
}

```

```

}
if (config_file[0] != '\0')
{
    /* A config file was passed as an option. Use it to populate the receiver topic
     * attributes.
     */
    rc = lbmaux_rcv_topic_attr_setopt_from_file(data->mTopicAttributes, config_file);
    if (rc == LBM_FAILURE)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),
                 "lbmaux_rcv_topic_attr_setopt_from_file() failed, %s",
                 lbm_errmsg());
        rcv_cleanup(data);
        return (-1);
    }
}
/* Go back through the options, looking for any specific receiver options. */
ptr = (const char *) TransportOptions;
while ((ptr = lbmon_next_key_value_pair(ptr, key, sizeof(key), value, sizeof(value))) != NULL)
{
    if (sscanf(key, "%[a-zA-Z_]%[a-zA-Z_]", scope, option) != 2)
    {
        continue;
    }
    if (strcasecmp(scope, "receiver") == 0)
    {
        rc = lbm_rcv_topic_attr_str_setopt(data->mTopicAttributes, option, value);
        if (rc == LBM_FAILURE)
        {
            snprintf(ErrorString,
                     sizeof(ErrorString),
                     "invalid option [receiver %s %s], %s",
                     option,
                     value,
                     lbm_errmsg());
            rcv_cleanup(data);
            return (rc);
        }
    }
}
entry = &ReceiverTopicOption[0];
while (entry->option != NULL)
{
    rc = lbm_rcv_topic_attr_str_setopt(data->mTopicAttributes, entry->option, entry->value);
    if (rc == LBM_FAILURE)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),
                 "error setting option [receiver %s %s], %s",
                 entry->option,
                 entry->value,
                 lbm_errmsg());
        rcv_cleanup(data);
        return (rc);
    }
    entry++;
}

```

```

/* For a non-wildcard topic, lookup the topic. */
if (wildcard_topic[0] == '\0')
{
    rc = lbm_rcv_topic_lookup(&(data->mTopic), data->mContext, topic, data->mTopicAttributes);
    if (rc == LBM_FAILURE)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),
                 "lbtm_rcv_topic_lookup() failed, %s",
                 lbm_errmsg());
        rcv_cleanup(data);
        return (rc);
    }
}

#ifndef _WIN32
    InitializeCriticalSection(&(data->mLock));
#else
    pthread_mutex_init(&(data->mLock), NULL);
#endif
data->mLockCreated = 1;
lock_receiver(data);
if (wildcard_topic[0] != '\0')
{
    /* Wildcard topic, create a wildcard receiver */
    rc = lbm_wildcard_rcv_create(&(data->mWildcardReceiver),
                                 data->mContext,
                                 wildcard_topic,
                                 data->mTopicAttributes,
                                 data->mWildcardReceiverAttributes,
                                 receive_callback,
                                 data,
                                 NULL);
}
else
{
    /* Non-wildcard topic, create a normal receiver */
    rc = lbm_rcv_create(&(data->mReceiver),
                        data->mContext,
                        data->mTopic,
                        receive_callback,
                        data,
                        NULL);
}
if (rc == LBM_FAILURE)
{
    snprintf(ErrorString,
             sizeof(ErrorString),
             "lbtm_wildcard_rcv_create()/lbtm_rcv_create() failed, %s",
             lbm_errmsg());
    unlock_receiver(data);
    rcv_cleanup(data);
    return (rc);
}

/* Pass back the lbmontrlbmsnmp_rcv_t created */
*TransportClientData = data;

```

```

        unlock_receiver(data);
        return (0);
    }

int
lbmmmon_transport_lbmsnmp_send(const char * Data, size_t Length, void * TransportClientData)
{
    lbmmmon_transport_lbmsnmp_src_t * src;
    int rc;

    if ((Data == NULL) || (TransportClientData == NULL))
    {
        strncpy(ErrorString, "Invalid argument", sizeof(ErrorString));
        return (-1);
    }
    src = (lbmmmon_transport_lbmsnmp_src_t *) TransportClientData;
    rc = lbm_src_send(src->mSource, Data, Length, 0);
    if (rc == LBM_FAILURE)
    {
        sprintf(ErrorString,
                sizeof(ErrorString),
                "lbt_src_send() failed, %s",
                lbm_errmsg());
    }
    return (rc);
}

int
lbmmmon_transport_lbmsnmp_receive(char * Data, size_t * Length, unsigned int TimeoutMS, void * TransportClientData)
{
    lbmmmon_transport_lbmsnmp_rcv_t * rcv = (lbmmmon_transport_lbmsnmp_rcv_t *) TransportClientData;
    lbmmmon_transport_lbmsnmp_rcv_node_t * node;
    int rc = 0;
    size_t length_remaining;
#if defined(_WIN32)
#elif defined(__TANDEM)
    unsigned int sleep_sec;
    unsigned int sleep_usec;
#else
    struct timespec ivl;
#endif

    if ((Data == NULL) || (Length == NULL) || (TransportClientData == NULL))
    {
        strncpy(ErrorString, "Invalid argument", sizeof(ErrorString));
        return (-1);
    }
    if (*Length == 0)
    {
        return (0);
    }
    lock_receiver(rcv);
    if (rcv->mHead != NULL)
    {
        /* Queue is non-empty. Pull the first message from the queue. */
        node = rcv->mHead;
        length_remaining = node->mMessage->len - node->mUsedBytes;
    }
}

```

```

        if (*Length >= length_remaining)
        {
            /* We can transfer the rest of the message */
            memcpy(Data, node->mMessage->data + node->mUsedBytes, length_remaining);
            *Length = length_remaining;
            rc = 0;
            /* We're done with the LBM message, so let LBM know. */
            lbm_msg_delete(node->mMessage);
            /* Unlink the node from the queue */
            rcv->mHead = node->mNext;
            if (rcv->mHead == NULL)
            {
                rcv->mTail = NULL;
            }
            free(node);
        }
        else
        {
            /* Can only transfer part of the message */
            memcpy(Data, node->mMessage->data + node->mUsedBytes, *Length);
            node->mUsedBytes -= *Length;
            rc = 0;
        }
        unlock_receiver(rcv);
    }
    else
    {
        unlock_receiver(rcv);
        /* Sleep for wait time */
#define NANOSECONDS_PER_SECOND 1000000000
#define MICROSECONDS_PER_SECOND 1000000
#define MILLISECONDS_PER_SECOND 1000
#define NANOSECONDS_PER_MILLISECOND (NANOSECONDS_PER_SECOND / MILLISECONDS_PER_SECOND)
#define MICROSECONDS_PER_MILLISECOND (MICROSECONDS_PER_SECOND / MILLISECONDS_PER_SECOND)
#if defined(_WIN32)
        Sleep(TimeoutMS);
#elif defined(__TANDEM)
        sleep_sec = TimeoutMS / MILLISECONDS_PER_SECOND;
        sleep_usec = (TimeoutMS % MILLISECONDS_PER_SECOND) * MICROSECONDS_PER_MILLISECOND;
        if (sleep_usec > 0)
        {
            usleep(sleep_usec);
        }
        if (sleep_sec > 0)
        {
            sleep(sleep_sec);
        }
#else
        ivl.tv_sec = TimeoutMS / MILLISECONDS_PER_SECOND;
        ivl.tv_nsec = (TimeoutMS % MILLISECONDS_PER_SECOND) * NANOSECONDS_PER_MILLISECOND;
        nanosleep(&ivl, NULL);
#endif
        rc = 1;
    }
    return (rc);
}

```

```

void
src_cleanup(lbmmon_transport_lbmsnmp_src_t * Data)
{
    if (Data->mSource != NULL)
    {
        lbm_src_delete(Data->mSource);
        Data->mSource = NULL;
    }
    Data->mTopic = NULL;
    if (Data->mTopicAttributes != NULL)
    {
        lbm_src_topic_attr_delete(Data->mTopicAttributes);
        Data->mTopicAttributes = NULL;
    }
    if (Data->mContext != NULL)
    {
        lbm_context_delete(Data->mContext);
        Data->mContext = NULL;
    }
    if (Data->mContextAttributes != NULL)
    {
        lbm_context_attr_delete(Data->mContextAttributes);
        Data->mContextAttributes = NULL;
    }
    free(Data);
}

int
lbmmmon_transport_lbmsnmp_src_finish(void * TransportClientData)
{
    lbmmon_transport_lbmsnmp_src_t * src;

    if (TransportClientData == NULL)
    {
        strncpy(ErrorString, "Invalid argument", sizeof(ErrorString));
        return (-1);
    }
    src = (lbmmon_transport_lbmsnmp_src_t *) TransportClientData;
    src_cleanup(src);
    return (0);
}

void
rcv_cleanup(lbmmon_transport_lbmsnmp_rcv_t * Data)
{
    lbmmon_transport_lbmsnmp_rcv_node_t * node;
    lbmmon_transport_lbmsnmp_rcv_node_t * next;

    /* Stop the receiver to prevent any more incoming messages */
    if (Data->mWildcardReceiver != NULL)
    {
        lbm_wildcard_rcv_delete(Data->mWildcardReceiver);
        Data->mWildcardReceiver = NULL;
    }
    if (Data->mWildcardReceiverAttributes != NULL)
    {
        lbm_wildcard_rcv_attr_delete(Data->mWildcardReceiverAttributes);
    }
}

```

```

        Data->mWildcardReceiverAttributes = NULL;
    }
    if (Data->mReceiver != NULL)
    {
        lbm_rcv_delete(Data->mReceiver);
        Data->mReceiver = NULL;
    }
    if (Data->mTopicAttributes != NULL)
    {
        lbm_rcv_topic_attr_delete(Data->mTopicAttributes);
        Data->mTopicAttributes = NULL;
    }
    Data->mTopic = NULL;

    /* Lock the receiver */
    if (Data->mLockCreated != 0)
    {
        lock_receiver(Data);
    }

    /* Delete the context to really make sure no more messages come in */
    if (Data->mContext != NULL)
    {
        lbm_context_delete(Data->mContext);
        Data->mContext = NULL;
    }
    if (Data->mContextAttributes != NULL)
    {
        lbm_context_attr_delete(Data->mContextAttributes);
        Data->mContextAttributes = NULL;
    }

    /* Clean out the queue */
    node = Data->mHead;
    while (node != NULL)
    {
        /* Let LBM know we're done with the message */
        lbm_msg_delete(node->mMessage);
        next = node->mNext;
        free(node);
        node = next;
    }

    if (Data->mLockCreated)
    {
        unlock_receiver(Data);
#endif _WIN32
        DeleteCriticalSection(&(Data->mLock));
#else
        pthread_mutex_destroy(&(Data->mLock));
#endif
    }

    free(Data);
}

int

```

```

lbmmmon_transport_lbmsnmp_rcv_finish(void * TransportClientData)
{
    lbmmmon_transport_lbmsnmp_rcv_t * rcv;

    if (TransportClientData == NULL)
    {
        strncpy(ErrorString, "Invalid argument", sizeof(ErrorString));
        return (-1);
    }
    rcv = (lbmmmon_transport_lbmsnmp_rcv_t *) TransportClientData;
    rcv_cleanup(rcv);
    return (0);
}

void
lock_receiver(lbmmmon_transport_lbmsnmp_rcv_t * Receiver)
{
#ifdef _WIN32
    EnterCriticalSection(&(Receiver->mLock));
#else
    pthread_mutex_lock(&(Receiver->mLock));
#endif
}

void
unlock_receiver(lbmmmon_transport_lbmsnmp_rcv_t * Receiver)
{
#ifdef _WIN32
    LeaveCriticalSection(&(Receiver->mLock));
#else
    pthread_mutex_unlock(&(Receiver->mLock));
#endif
}

const char *
lbmmmon_transport_lbmsnmp_errmsg(void)
{
    return (ErrorString);
}

int
scope_is_valid(const char * Scope)
{
    if (strcasecmp(Scope, "context") == 0)
    {
        return (0);
    }
    if (strcasecmp(Scope, "source") == 0)
    {
        return (0);
    }
    if (strcasecmp(Scope, "receiver") == 0)
    {
        return (0);
    }
    if (strcasecmp(Scope, "event_queue") == 0)
    {

```

```
        return (0);
    }
    return (-1);
}
```

## 9.14 Deprecated List

Class [lbtmon\\_msg\\_gateway\\_info\\_t\\_stct](#)

Global [lbtmon\\_msg\\_retrieve\\_gateway\\_info](#)

Global [LBMMON\\_ATTR\\_CONTEXTID](#) Use [LBMMON\\_ATTR\\_OBJECTID](#) instead.

Global [lbmon\\_attr\\_get\\_contextid](#) Use [lbmon\\_attr\\_get\\_objectid](#) instead.

# Index

add  
    lbmsdm\_msg\_add\_blob, 16  
    lbmsdm\_msg\_add\_boolean, 16  
    lbmsdm\_msg\_add\_decimal, 17  
    lbmsdm\_msg\_add\_double, 17  
    lbmsdm\_msg\_add\_float, 17  
    lbmsdm\_msg\_add\_int16, 17  
    lbmsdm\_msg\_add\_int32, 17  
    lbmsdm\_msg\_add\_int64, 17  
    lbmsdm\_msg\_add\_int8, 18  
    lbmsdm\_msg\_add\_message, 18  
    lbmsdm\_msg\_add\_string, 18  
    lbmsdm\_msg\_add\_timestamp, 18  
    lbmsdm\_msg\_add\_uint16, 18  
    lbmsdm\_msg\_add\_uint32, 18  
    lbmsdm\_msg\_add\_uint64, 18  
    lbmsdm\_msg\_add\_uint8, 19  
    lbmsdm\_msg\_add\_unicode, 19  
Add a field to a message, 15  
Add an array field to a message, 20  
Add an element to an array field by field index, 24  
Add an element to an array field by field name, 29  
Add an element to an array field referenced by an iterator, 34  
add\_array  
    lbmsdm\_msg\_add\_blob\_array, 21  
    lbmsdm\_msg\_add\_boolean\_array, 21  
    lbmsdm\_msg\_add\_decimal\_array, 21  
    lbmsdm\_msg\_add\_double\_array, 21  
    lbmsdm\_msg\_add\_float\_array, 21  
    lbmsdm\_msg\_add\_int16\_array, 21  
    lbmsdm\_msg\_add\_int32\_array, 21  
    lbmsdm\_msg\_add\_int64\_array, 22  
lbmsdm\_msg\_add\_int8\_array, 22  
lbmsdm\_msg\_add\_message\_array, 22  
lbmsdm\_msg\_add\_string\_array, 22  
lbmsdm\_msg\_add\_timestamp\_array, 22  
lbmsdm\_msg\_add\_uint16\_array, 22  
lbmsdm\_msg\_add\_uint32\_array, 22  
lbmsdm\_msg\_add\_uint64\_array, 23  
lbmsdm\_msg\_add\_uint8\_array, 23  
lbmsdm\_msg\_add\_unicode\_array, 23  
add\_elem\_idx  
    lbmsdm\_msg\_add\_blob\_elem\_idx, 25  
    lbmsdm\_msg\_add\_boolean\_elem\_idx, 25  
    lbmsdm\_msg\_add\_decimal\_elem\_idx, 25  
    lbmsdm\_msg\_add\_double\_elem\_idx, 25  
    lbmsdm\_msg\_add\_float\_elem\_idx, 26  
    lbmsdm\_msg\_add\_int16\_elem\_idx, 26  
    lbmsdm\_msg\_add\_int32\_elem\_idx, 26  
    lbmsdm\_msg\_add\_int64\_elem\_idx, 26  
    lbmsdm\_msg\_add\_int8\_elem\_idx, 26  
    lbmsdm\_msg\_add\_message\_elem\_idx, 26  
    lbmsdm\_msg\_add\_string\_elem\_idx, 26  
    lbmsdm\_msg\_add\_timestamp\_elem\_idx, 27

**lbmsdm\_msg\_add\_uint16\_elem\_idx**, 27  
**lbmsdm\_msg\_add\_uint32\_elem\_idx**, 27  
**lbmsdm\_msg\_add\_uint64\_elem\_idx**, 27  
**lbmsdm\_msg\_add\_uint8\_elem\_idx**, 27  
**lbmsdm\_msg\_add\_unicode\_elem\_idx**, 27  
**add\_elem\_iter**  
    **lbmsdm\_iter\_add\_blob\_elem**, 35  
    **lbmsdm\_iter\_add\_boolean\_elem**, 35  
    **lbmsdm\_iter\_add\_decimal\_elem**, 35  
    **lbmsdm\_iter\_add\_double\_elem**, 35  
    **lbmsdm\_iter\_add\_float\_elem**, 36  
    **lbmsdm\_iter\_add\_int16\_elem**, 36  
    **lbmsdm\_iter\_add\_int32\_elem**, 36  
    **lbmsdm\_iter\_add\_int64\_elem**, 36  
    **lbmsdm\_iter\_add\_int8\_elem**, 36  
    **lbmsdm\_iter\_add\_message\_elem**, 36  
    **lbmsdm\_iter\_add\_string\_elem**, 36  
    **lbmsdm\_iter\_add\_timestamp\_elem**, 37  
    **lbmsdm\_iter\_add\_uint16\_elem**, 37  
    **lbmsdm\_iter\_add\_uint32\_elem**, 37  
    **lbmsdm\_iter\_add\_uint64\_elem**, 37  
    **lbmsdm\_iter\_add\_uint8\_elem**, 37  
    **lbmsdm\_iter\_add\_unicode\_elem**, 37  
**add\_elem\_name**  
    **lbmsdm\_msg\_add\_blob\_elem\_name**, 30  
    **lbmsdm\_msg\_add\_boolean\_elem\_name**, 30  
    **lbmsdm\_msg\_add\_decimal\_elem\_name**, 30  
    **lbmsdm\_msg\_add\_double\_elem\_name**, 30  
    **lbmsdm\_msg\_add\_float\_elem\_name**, 31  
    **lbmsdm\_msg\_add\_int16\_elem\_name**, 31  
    **lbmsdm\_msg\_add\_int32\_elem\_name**, 31  
**lbmsdm\_msg\_add\_int64\_elem\_name**, 31  
**lbmsdm\_msg\_add\_int8\_elem\_name**, 31  
**lbmsdm\_msg\_add\_message\_elem\_name**, 31  
**lbmsdm\_msg\_add\_string\_elem\_name**, 31  
**lbmsdm\_msg\_add\_timestamp\_elem\_name**, 32  
**lbmsdm\_msg\_add\_uint16\_elem\_name**, 32  
**lbmsdm\_msg\_add\_uint32\_elem\_name**, 32  
**lbmsdm\_msg\_add\_uint64\_elem\_name**, 32  
**lbmsdm\_msg\_add\_uint8\_elem\_name**, 32  
**lbmsdm\_msg\_add\_unicode\_elem\_name**, 32  
**addr**  
    **lbm\_ipv4\_address\_mask\_t\_stct**, 147  
**age\_max**  
    **lbm\_event\_queue\_stats\_t\_stct**, 142  
**age\_mean**  
    **lbm\_event\_queue\_stats\_t\_stct**, 141  
**age\_min**  
    **lbm\_event\_queue\_stats\_t\_stct**, 141  
**apphdr\_chain**  
    **lbm\_src\_send\_ex\_info\_t\_stct**, 222  
**application\_set\_index**  
    **lbm\_src\_event\_umq\_ulb\_message\_info\_ex\_t\_stct**, 216  
    **lbm\_src\_event\_umq\_ulb\_receiver\_info\_ex\_t\_stct**, 217  
    **lbm\_umq\_ulb\_receiver\_type\_entry\_t\_stct**, 267  
**appname**  
    **lbm\_umm\_info\_t\_stct**, 255  
**appsets**  
    **lbm\_umq\_queue\_topic\_t\_stct**, 264  
**assignment\_id**  
    **lbm\_msg\_umq\_registration\_complete\_ex\_t\_stct**, 169  
    **lbm\_src\_event\_umq\_ulb\_message\_info\_ex\_t\_stct**, 216

lbm\_src\_event\_umq\_ulb\_receiver\_info\_ex\_t\_stct, 217  
async\_opfunc  
  lbm\_src\_send\_ex\_info\_t\_stct, 222  
  
bits  
  lbm\_ipv4\_address\_mask\_t\_stct, 147  
bytes  
  lbm\_flight\_size\_inflight\_t\_stct, 144  
bytes\_buffered  
  lbm\_src\_transport\_stats\_daemon\_t\_stct, 223  
  lbm\_src\_transport\_stats\_tcp\_t\_stct, 235  
bytes\_rcv  
  lbm\_rcv\_transport\_stats\_daemon\_t\_stct, 172  
  lbm\_rcv\_transport\_stats\_lbtpc\_t\_stct, 173  
  lbm\_rcv\_transport\_stats\_lbtrdma\_t\_stct, 175  
  lbm\_rcv\_transport\_stats\_lbtrm\_t\_stct, 178  
  lbm\_rcv\_transport\_stats\_lbtru\_t\_stct, 185  
  lbm\_rcv\_transport\_stats\_tcp\_t\_stct, 192  
bytes\_sent  
  lbm\_src\_transport\_stats\_lbtpc\_t\_stct, 224  
  lbm\_src\_transport\_stats\_lbtrdma\_t\_stct, 225  
  lbm\_src\_transport\_stats\_lbtrm\_t\_stct, 226  
  lbm\_src\_transport\_stats\_lbtru\_t\_stct, 230  
  
callback\_events  
  lbm\_event\_queue\_stats\_t\_stct, 142  
callback\_events\_svc\_max  
  lbm\_event\_queue\_stats\_t\_stct, 142  
callback\_events\_svc\_mean  
  lbm\_event\_queue\_stats\_t\_stct, 142  
callback\_events\_svc\_min  
  lbm\_event\_queue\_stats\_t\_stct, 142  
callback\_events\_tot  
  lbm\_event\_queue\_stats\_t\_stct, 142  
  
cancel\_events  
  lbm\_event\_queue\_stats\_t\_stct, 139  
cancel\_events\_svc\_max  
  lbm\_event\_queue\_stats\_t\_stct, 140  
cancel\_events\_svc\_mean  
  lbm\_event\_queue\_stats\_t\_stct, 140  
cancel\_events\_svc\_min  
  lbm\_event\_queue\_stats\_t\_stct, 140  
cancel\_events\_tot  
  lbm\_event\_queue\_stats\_t\_stct, 139  
cbfunc  
  lbmon\_ctx\_statistics\_func\_t\_stct, 273  
  lbmon\_evq\_statistics\_func\_t\_stct, 274  
  lbmon\_rcv\_statistics\_func\_t\_stct, 279  
  lbmon\_src\_statistics\_func\_t\_stct, 280  
cbproc  
  lbm\_delete\_cb\_info\_t\_stct, 130  
  lbm\_event\_queue\_cancel\_cb\_info\_t\_stct, 131  
cert\_file  
  lbm\_umm\_info\_t\_stct, 256  
cert\_file\_password  
  lbm\_umm\_info\_t\_stct, 256  
channel\_info  
  lbm\_msg\_t\_stct, 156  
  lbm\_src\_send\_ex\_info\_t\_stct, 221  
channel\_number  
  lbm\_msg\_channel\_info\_t\_stct, 149  
clientd  
  lbm\_async\_operation\_func\_t, 115  
  lbm\_delete\_cb\_info\_t\_stct, 130  
  lbm\_event\_queue\_cancel\_cb\_info\_t\_stct, 131  
  lbm\_umq\_queue\_msg\_status\_t, 261  
context\_source\_events  
  lbm\_event\_queue\_stats\_t\_stct, 140  
context\_source\_events\_svc\_max  
  lbm\_event\_queue\_stats\_t\_stct, 141  
context\_source\_events\_svc\_mean  
  lbm\_event\_queue\_stats\_t\_stct, 141  
context\_source\_events\_svc\_min

lbm\_event\_queue\_stats\_t\_stct, 140  
 context\_source\_events\_tot  
     lbm\_event\_queue\_stats\_t\_stct, 140  
 copied\_state  
     lbm\_msg\_t\_stct, 154

d

lbm\_umq\_ulb\_application\_set\_-  
     attr\_t\_stct, 265  
 lbm\_umq\_ulb\_receiver\_type\_attr\_-  
     t\_stct, 266

daemon

lbm\_rcv\_transport\_stats\_t\_stct, 191  
 lbm\_src\_transport\_stats\_t\_stct, 234

data

lbm\_apphdr\_chain\_elem\_t\_stct, 114  
 lbm\_msg\_t\_stct, 156

data\_msgs

lbm\_event\_queue\_stats\_t\_stct, 133

data\_msgs\_svc\_max

lbm\_event\_queue\_stats\_t\_stct, 134

data\_msgs\_svc\_mean

lbm\_event\_queue\_stats\_t\_stct, 134

data\_msgs\_svc\_min

lbm\_event\_queue\_stats\_t\_stct, 133

data\_msgs\_tot

lbm\_event\_queue\_stats\_t\_stct, 133

dest\_port

lbm\_transport\_source\_info\_t\_stct,  
     239

destination\_port

lbm\_icast\_resolver\_entry\_t\_stct,  
     241

dgrams\_dropped\_hdr

lbm\_rcv\_transport\_stats\_lbtrm\_t\_-  
     stct, 182  
 lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
     stct, 189

dgrams\_dropped\_other

lbm\_rcv\_transport\_stats\_lbtrm\_t\_-  
     stct, 182  
 lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
     stct, 189

dgrams\_dropped\_sid

lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
     stct, 189

dgrams\_dropped\_size  
     lbm\_rcv\_transport\_stats\_lbtrm\_t\_-  
         stct, 182  
     lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
         stct, 188

dgrams\_dropped\_type

lbm\_rcv\_transport\_stats\_lbtrm\_t\_-  
     stct, 182  
 lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
     stct, 189

dgrams\_dropped\_version

lbm\_rcv\_transport\_stats\_lbtrm\_t\_-  
     stct, 182  
 lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
     stct, 189

duplicate\_data

lbm\_rcv\_transport\_stats\_lbtrm\_t\_-  
     stct, 180  
 lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
     stct, 187

event\_queue

lbm\_event\_queue\_cancel\_cb\_info\_-  
     t\_stct, 131

events

lbm\_event\_queue\_stats\_t\_stct, 141

events\_tot

lbm\_event\_queue\_stats\_t\_stct, 141

evq

lbm\_async\_operation\_func\_t, 115

exp

lbmpdm\_decimal\_t, 283  
 lbmsdm\_decimal\_t\_stct, 288

field\_type

lbmpdm\_field\_value\_stct\_t, 285

first\_sequence\_number

lbm\_src\_event\_sequence\_number\_-  
     info\_t\_stct, 199  
 lbm\_src\_event\_umq\_stability\_ack\_-  
     info\_ex\_t\_stct, 213  
 lbm\_src\_event\_umq\_ulb\_message\_-  
     info\_ex\_t\_stct, 216

fixed\_str\_len

lbmpdm\_field\_info\_attr\_stct\_t, 284

flags

lbm\_async\_operation\_func\_t, 115  
lbm\_async\_operation\_info\_t, 117  
lbm\_context\_event\_umq\_-  
    registration\_complete\_ex\_-  
        t\_stct, 119  
lbm\_context\_event\_umq\_-  
    registration\_ex\_t\_stct, 121  
lbm\_msg\_channel\_info\_t\_stct, 149  
lbm\_msg\_t\_stct, 155  
lbm\_msg\_ume\_deregistration\_ex\_-  
    t\_stct, 158  
lbm\_msg\_ume\_registration\_-  
    complete\_ex\_t\_stct, 160  
lbm\_msg\_ume\_registration\_ex\_t\_-  
    stct, 161  
lbm\_msg\_umq\_deregistration\_-  
    complete\_ex\_t\_stct, 164  
lbm\_msg\_umq\_index\_assigned\_-  
    ex\_t\_stct, 165  
lbm\_msg\_umq\_index\_assignment\_-  
    eligibility\_start\_complete\_ex\_-  
        t\_stct, 166  
lbm\_msg\_umq\_index\_assignment\_-  
    eligibility\_stop\_complete\_ex\_-  
        t\_stct, 167  
lbm\_msg\_umq\_index\_released\_ex\_-  
    t\_stct, 168  
lbm\_msg\_umq\_registration\_-  
    complete\_ex\_t\_stct, 169  
lbm\_src\_event\_sequence\_number\_-  
    info\_t\_stct, 199  
lbm\_src\_event\_ume\_ack\_ex\_info\_-  
    t\_stct, 201  
lbm\_src\_event\_ume\_-  
    deregistration\_ex\_t\_stct,  
        204  
lbm\_src\_event\_ume\_registration\_-  
    complete\_ex\_t\_stct, 206  
lbm\_src\_event\_ume\_registration\_-  
    ex\_t\_stct, 207  
lbm\_src\_event\_umq\_message\_id\_-  
    info\_t\_stct, 210  
lbm\_src\_event\_umq\_registration\_-  
    complete\_ex\_t\_stct, 212  
lbm\_src\_event\_umq\_stability\_ack\_-  
    info\_ex\_t\_stct, 213  
lbm\_src\_event\_umq\_ulb\_message\_-  
    info\_ex\_t\_stct, 215  
lbm\_src\_event\_umq\_ulb\_receiver\_-  
    info\_ex\_t\_stct, 217  
lbm\_src\_event\_wakeup\_t\_stct, 219  
lbm\_src\_send\_ex\_info\_t\_stct, 221  
lbm\_ume\_rcv\_recovery\_info\_ex\_-  
    func\_info\_t\_stct, 244  
lbm\_ume\_rcv\_regid\_ex\_func\_info\_-  
    t\_stct, 247  
lbm\_umm\_info\_t\_stct, 255  
lbm\_umq\_index\_info\_t\_stct, 257  
lbm\_umq\_msg\_total\_lifetime\_-  
    info\_t\_stct, 258  
lbm\_umq\_queue\_msg\_status\_t, 262  
lbm\_umq\_queue\_topic\_status\_t, 263  
func  
    lbm\_async\_operation\_func\_t, 115  
Get a scalar field via an iterator, 49  
Get an element from an array field by field  
    index, 54  
Get an element from an array field by field  
    name, 59  
Get an element from an array field refer-  
    enced by an iterator, 65  
Get scalar field values by field index, 39  
Get scalar field values by field name, 44  
get\_elem\_idx  
    lbmsdm\_msg\_get\_blob\_elem\_idx,  
        55  
    lbmsdm\_msg\_get\_boolean\_elem\_-  
        idx, 55  
    lbmsdm\_msg\_get\_decimal\_elem\_-  
        idx, 55  
    lbmsdm\_msg\_get\_double\_elem\_-  
        idx, 56  
    lbmsdm\_msg\_get\_float\_elem\_idx,  
        56  
    lbmsdm\_msg\_get\_int16\_elem\_idx,  
        56  
    lbmsdm\_msg\_get\_int32\_elem\_idx,  
        56  
    lbmsdm\_msg\_get\_int64\_elem\_idx,  
        56

lbmsdm\_msg\_get\_int8\_elem\_idx,  
     56  
 lbmsdm\_msg\_get\_message\_elem\_idx,  
     56  
 lbmsdm\_msg\_get\_string\_elem\_idx,  
     57  
 lbmsdm\_msg\_get\_timestamp\_elem\_idx,  
     57  
 lbmsdm\_msg\_get\_uint16\_elem\_idx,  
     57  
 lbmsdm\_msg\_get\_uint32\_elem\_idx,  
     57  
 lbmsdm\_msg\_get\_uint64\_elem\_idx,  
     58  
 lbmsdm\_msg\_get\_uint8\_elem\_idx,  
     58  
 lbmsdm\_msg\_get\_unicode\_elem\_idx,  
     58  
 get\_elem\_iter  
     lbmsdm\_iter\_get\_blob\_elem, 66  
     lbmsdm\_iter\_get\_boolean\_elem, 66  
     lbmsdm\_iter\_get\_decimal\_elem, 66  
     lbmsdm\_iter\_get\_double\_elem, 67  
     lbmsdm\_iter\_get\_float\_elem, 67  
     lbmsdm\_iter\_get\_int16\_elem, 67  
     lbmsdm\_iter\_get\_int32\_elem, 67  
     lbmsdm\_iter\_get\_int64\_elem, 67  
     lbmsdm\_iter\_get\_int8\_elem, 67  
     lbmsdm\_iter\_get\_message\_elem, 67  
     lbmsdm\_iter\_get\_string\_elem, 68  
     lbmsdm\_iter\_get\_timestamp\_elem,  
         68  
     lbmsdm\_iter\_get\_uint16\_elem, 68  
     lbmsdm\_iter\_get\_uint32\_elem, 68  
     lbmsdm\_iter\_get\_uint64\_elem, 68  
     lbmsdm\_iter\_get\_uint8\_elem, 69  
     lbmsdm\_iter\_get\_unicode\_elem, 69  
 get\_elem\_name  
     lbmsdm\_msg\_get\_blob\_elem\_name,  
         60  
     lbmsdm\_msg\_get\_boolean\_elem\_name,  
         60  
     lbmsdm\_msg\_get\_decimal\_elem\_name,  
         60  
     lbmsdm\_msg\_get\_double\_elem\_name,  
         61  
 lbmsdm\_msg\_get\_float\_elem\_name,  
     61  
 lbmsdm\_msg\_get\_int16\_elem\_name,  
     61  
 lbmsdm\_msg\_get\_int32\_elem\_name,  
     61  
 lbmsdm\_msg\_get\_int64\_elem\_name,  
     61  
 lbmsdm\_msg\_get\_int8\_elem\_name,  
     61  
 lbmsdm\_msg\_get\_message\_elem\_name,  
     62  
 lbmsdm\_msg\_get\_string\_elem\_name,  
     62  
 lbmsdm\_msg\_get\_timestamp\_elem\_name,  
     62  
 lbmsdm\_msg\_get\_uint16\_elem\_name,  
     62  
 lbmsdm\_msg\_get\_uint32\_elem\_name,  
     63  
 lbmsdm\_msg\_get\_uint64\_elem\_name,  
     63  
 lbmsdm\_msg\_get\_uint8\_elem\_name,  
     63  
 lbmsdm\_msg\_get\_unicode\_elem\_name,  
     63  
 get\_scalar\_idx  
     lbmsdm\_msg\_get\_blob\_idx, 40  
     lbmsdm\_msg\_get\_boolean\_idx, 40  
     lbmsdm\_msg\_get\_decimal\_idx, 40  
     lbmsdm\_msg\_get\_double\_idx, 41  
     lbmsdm\_msg\_get\_float\_idx, 41  
     lbmsdm\_msg\_get\_int16\_idx, 41  
     lbmsdm\_msg\_get\_int32\_idx, 41  
     lbmsdm\_msg\_get\_int64\_idx, 41  
     lbmsdm\_msg\_get\_int8\_idx, 41  
     lbmsdm\_msg\_get\_message\_idx, 41  
     lbmsdm\_msg\_get\_string\_idx, 42  
     lbmsdm\_msg\_get\_timestamp\_idx,  
         42  
     lbmsdm\_msg\_get\_uint16\_idx, 42  
     lbmsdm\_msg\_get\_uint32\_idx, 42  
     lbmsdm\_msg\_get\_uint64\_idx, 42  
     lbmsdm\_msg\_get\_uint8\_idx, 43  
     lbmsdm\_msg\_get\_unicode\_idx, 43  
 get\_scalar\_iter

lbmsdm\_iter\_get\_blob, 50  
lbmsdm\_iter\_get\_boolean, 50  
lbmsdm\_iter\_get\_decimal, 50  
lbmsdm\_iter\_get\_double, 50  
lbmsdm\_iter\_get\_float, 51  
lbmsdm\_iter\_get\_int16, 51  
lbmsdm\_iter\_get\_int32, 51  
lbmsdm\_iter\_get\_int64, 51  
lbmsdm\_iter\_get\_int8, 51  
lbmsdm\_iter\_get\_message, 51  
lbmsdm\_iter\_get\_string, 51  
lbmsdm\_iter\_get\_timestamp, 52  
lbmsdm\_iter\_get\_uint16, 52  
lbmsdm\_iter\_get\_uint32, 52  
lbmsdm\_iter\_get\_uint64, 52  
lbmsdm\_iter\_get\_uint8, 52  
lbmsdm\_iter\_get\_unicode, 53

get\_scalar\_name  
  lbmsdm\_msg\_get\_blob\_name, 45  
  lbmsdm\_msg\_get\_boolean\_name, 45  
  lbmsdm\_msg\_get\_decimal\_name, 45  
  lbmsdm\_msg\_get\_double\_name, 46  
  lbmsdm\_msg\_get\_float\_name, 46  
  lbmsdm\_msg\_get\_int16\_name, 46  
  lbmsdm\_msg\_get\_int32\_name, 46  
  lbmsdm\_msg\_get\_int64\_name, 46  
  lbmsdm\_msg\_get\_int8\_name, 46  
  lbmsdm\_msg\_get\_message\_name, 46  
  lbmsdm\_msg\_get\_string\_name, 47  
  lbmsdm\_msg\_get\_timestamp\_name, 47  
  lbmsdm\_msg\_get\_uint16\_name, 47  
  lbmsdm\_msg\_get\_uint32\_name, 47  
  lbmsdm\_msg\_get\_uint64\_name, 47  
  lbmsdm\_msg\_get\_uint8\_name, 48  
  lbmsdm\_msg\_get\_unicode\_name, 48

group\_index  
  lbum\_ume\_store\_entry\_t\_stct, 252  
  lbum\_ume\_store\_name\_entry\_t\_stct, 254

group\_size

lbum\_ume\_store\_group\_entry\_t\_stct, 253

handle  
  lbum\_async\_operation\_info\_t, 117

hashfunc  
  lbum\_str\_hash\_func\_ex\_t\_stct, 236

hf\_sequence\_number  
  lbum\_msg\_t\_stct, 157

hf\_sqn  
  lbum\_src\_send\_ex\_info\_t\_stct, 222

high\_sequence\_number  
  lbum\_ume\_rcv\_recovery\_info\_ex\_func\_info\_t\_stct, 245

id  
  lbum\_umq\_ulb\_receiver\_type\_attr\_t\_stct, 266  
  lbum\_umq\_ulb\_receiver\_type\_entry\_t\_stct, 267

iface  
  lbum\_icast\_resolver\_entry\_t\_stct, 241

index  
  lbum\_ume\_store\_group\_entry\_t\_stct, 253  
  lbum\_umq\_index\_info\_t\_stct, 257  
  lbum\_umq\_ulb\_application\_set\_attr\_t\_stct, 265

index\_len  
  lbum\_umq\_index\_info\_t\_stct, 257

info  
  lbum\_async\_operation\_info\_t, 117

io\_events  
  lbum\_event\_queue\_stats\_t\_stct, 137

io\_events\_svc\_max  
  lbum\_event\_queue\_stats\_t\_stct, 137

io\_events\_svc\_mean  
  lbum\_event\_queue\_stats\_t\_stct, 137

io\_events\_svc\_min  
  lbum\_event\_queue\_stats\_t\_stct, 137

io\_events\_tot  
  lbum\_event\_queue\_stats\_t\_stct, 137

iov\_base  
  lbum\_iovec\_t\_stct, 146

iov\_len

lbm\_iovec\_t\_stct, 146  
 ip\_address  
     lbm\_ume\_store\_entry\_t\_stct, 252  
 is\_array  
     lbmpdm\_field\_value\_stct\_t, 285  
 is\_fixed  
     lbmpdm\_field\_value\_stct\_t, 285  
  
 last\_sequence\_number  
     lbm\_src\_event\_sequence\_number\_info\_t\_stct, 199  
     lbm\_src\_event\_umq\_stability\_ack\_info\_ex\_t\_stct, 214  
     lbm\_src\_event\_umq\_ulb\_message\_info\_ex\_t\_stct, 216  
  
 lbm.h, 291  
     lbm\_apphdr\_chain\_append\_elem, 412  
     lbm\_apphdr\_chain\_create, 412  
     lbm\_apphdr\_chain\_delete, 412  
     lbm\_apphdr\_chain\_iter\_create, 412  
     lbm\_apphdr\_chain\_iter\_create\_from\_msg, 413  
     lbm\_apphdr\_chain\_iter\_current, 413  
     lbm\_apphdr\_chain\_iter\_delete, 413  
     lbm\_apphdr\_chain\_iter\_done, 413  
     lbm\_apphdr\_chain\_iter\_first, 414  
     lbm\_apphdr\_chain\_iter\_next, 414  
     LBM\_ASYNC\_OP\_INFO\_FLAG\_FIRST, 356  
     LBM\_ASYNC\_OP\_INFO\_FLAG\_INLINE, 356  
     LBM\_ASYNC\_OP\_INFO\_FLAG\_LAST, 356  
     LBM\_ASYNC\_OP\_INFO\_FLAG\_ONLY, 356  
     LBM\_ASYNC\_OP\_INVALID\_HANDLE, 356  
     LBM\_ASYNC\_OP\_STATUS\_CANCELED, 356  
     LBM\_ASYNC\_OP\_STATUS\_COMPLETE, 356  
     LBM\_ASYNC\_OP\_STATUS\_ERROR, 356  
     LBM\_ASYNC\_OP\_STATUS\_IN\_PROGRESS, 356  
  
 LBM\_ASYNC\_OP\_TYPE\_CTX\_UMQ\_QUEUE\_TOPIC\_LIST, 357  
 LBM\_ASYNC\_OP\_TYPE\_RCV\_UMQ\_QUEUE\_MSG\_LIST, 357  
 LBM\_ASYNC\_OP\_TYPE\_RCV\_UMQ\_QUEUE\_MSG\_RETRIEVE, 357  
 lbm\_async\_operation\_cancel, 414  
 LBM\_ASYNC\_OPERATION\_CANCEL\_FLAG\_NONBLOCK, 357  
 lbm\_async\_operation\_function\_cb, 385  
 lbm\_async\_operation\_status, 415  
 LBM\_ASYNC\_OPERATION\_STATUS\_FLAG\_NONBLOCK, 357  
 lbm\_auth\_set\_credentials, 416  
 lbm\_authstorage\_addtpnam, 416  
 lbm\_authstorage\_checkpermission, 417  
 lbm\_authstorage\_close\_storage\_xml, 417  
 lbm\_authstorage\_deltpnam, 417  
 lbm\_authstorage\_load\_roletable, 417  
 lbm\_authstorage\_open\_storage\_xml, 418  
 lbm\_authstorage\_print\_roletable, 418  
 lbm\_authstorage\_roletable\_add\_role\_action, 418  
 lbm\_authstorage\_unload\_roletable, 419  
 lbm\_authstorage\_user\_add\_role, 419  
 lbm\_authstorage\_user\_del\_role, 419  
 lbm\_cancel\_fd, 420  
 lbm\_cancel\_fd\_ex, 420  
 lbm\_cancel\_timer, 421  
 lbm\_cancel\_timer\_ex, 422  
 LBM\_CHAIN\_ELEM\_APPHDR, 357

LBM\_CHAIN\_ELEM\_-  
    CHANNEL\_NUMBER,  
        357  
LBM\_CHAIN\_ELEM\_GW\_INFO,  
    357  
LBM\_CHAIN\_ELEM\_HF\_SQN,  
    358  
LBM\_CHAIN\_ELEM\_-  
    PROPERTIES\_LENGTH,  
        358  
LBM\_CHAIN\_ELEM\_USER\_-  
    DATA, 358  
lbm\_config, 422  
lmb\_config\_xml\_file, 422  
lmb\_config\_xml\_string, 423  
lmb\_context\_attr\_create, 423  
lmb\_context\_attr\_create\_default,  
    424  
lmb\_context\_attr\_create\_from\_xml,  
    424  
lmb\_context\_attr\_delete, 424  
lmb\_context\_attr\_dump, 425  
lmb\_context\_attr\_dup, 425  
lmb\_context\_attr\_getopt, 425  
lmb\_context\_attr\_option\_size, 426  
lmb\_context\_attr\_set\_from\_xml,  
    426  
lmb\_context\_attr\_setopt, 426  
lmb\_context\_attr\_str\_getopt, 427  
lmb\_context\_attr\_str\_setopt, 427  
lmb\_context\_create, 428  
lmb\_context\_delete, 428  
lmb\_context\_delete\_ex, 429  
lmb\_context\_dump, 429  
lmb\_context\_event\_cb\_proc, 385  
lmb\_context\_event\_func\_t, 386  
LBM\_CONTEXT\_EVENT\_-  
    UMQ\_INSTANCE\_LIST\_-  
        NOTIFICATION, 358  
LBM\_CONTEXT\_EVENT\_-  
    UMQ\_REGISTRATION\_-  
        COMPLETE\_EX, 358  
LBM\_CONTEXT\_EVENT\_-  
    UMQ\_REGISTRATION\_-  
        COMPLETE\_EX\_FLAG\_-  
            QUORUM, 358  
lmb\_context\_event\_umq\_-  
    registration\_complete\_ex\_t,  
        386  
LBM\_CONTEXT\_EVENT\_-  
    UMQ\_REGISTRATION\_-  
        ERROR, 358  
lmb\_context\_event\_umq\_-  
    registration\_ex\_t, 386  
LBM\_CONTEXT\_EVENT\_-  
    UMQ\_REGISTRATION\_-  
        SUCCESS\_EX, 358  
lmb\_context\_from\_rcv, 429  
lmb\_context\_from\_src, 429  
lmb\_context\_from\_wildcard\_rcv,  
    430  
lmb\_context\_get\_name, 430  
lmb\_context\_getopt, 430  
lmb\_context\_lbtpc\_unblock, 431  
lmb\_context\_process\_events, 431  
lmb\_context\_process\_lbtpc\_-  
    messages, 431  
lmb\_context\_rcv\_immediate\_msgs,  
    432  
lmb\_context\_rcv\_immediate\_-  
    msgs\_func\_t, 386  
lmb\_context\_rcv\_immediate\_-  
    topic\_msgs, 432  
lmb\_context\_reactor\_only\_create,  
    433  
lmb\_context\_reset\_im\_rcv\_-  
    transport\_stats, 433  
lmb\_context\_reset\_im\_src\_-  
    transport\_stats, 434  
lmb\_context\_reset\_rcv\_transport\_-  
    stats, 434  
lmb\_context\_reset\_src\_transport\_-  
    stats, 434  
lmb\_context\_reset\_stats, 434  
lmb\_context\_retrieve\_im\_rcv\_-  
    transport\_stats, 434  
lmb\_context\_retrieve\_im\_src\_-  
    transport\_stats, 435  
lmb\_context\_retrieve\_rcv\_-  
    transport\_stats, 435  
lmb\_context\_retrieve\_src\_-  
    transport\_stats, 436

lbm\_context\_retrieve\_stats, 436  
 lbm\_context\_set\_name, 437  
 lbm\_context\_setopt, 437  
 lbm\_context\_src\_cb\_proc, 386  
 lbm\_context\_src\_event\_func\_t, 387  
 lbm\_context\_stats\_t, 387  
 lbm\_context\_str\_getopt, 437  
 lbm\_context\_str\_setopt, 438  
 lbm\_context\_topic\_resolution\_request, 438  
 lbm\_context\_unblock, 439  
 lbm\_create\_random\_id, 439  
 lbm\_ctx\_umq\_get\_inflight, 439  
 lbm\_ctx\_umq\_queue\_topic\_list, 440  
 lbm\_daemon\_event\_cb\_proc, 387  
 LBM\_DAEMON\_EVENT\_CONNECT\_ERROR, 359  
 LBM\_DAEMON\_EVENT\_CONNECT\_TIMEOUT, 359  
 LBM\_DAEMON\_EVENT\_CONNECTED, 359  
 LBM\_DAEMON\_EVENT\_DISCONNECTED, 359  
 lbm\_debug\_dump, 440  
 lbm\_debug\_filename, 440  
 lbm\_debug\_mask, 441  
 lbm\_deserialize\_response, 441  
 LBM\_EDAEMONCONN, 359  
 LBM\_EINPROGRESS, 359  
 LBM\_EINVAL, 359  
 LBM\_EMSG\_SELECTOR, 359  
 LBM\_ENO\_QUEUE\_REG, 359  
 LBM\_ENO\_STORE\_REG, 360  
 LBM\_ENOMEM, 360  
 LBM\_EOP, 360  
 LBM\_EOPNOTSUPP, 360  
 LBM\_EOS, 360  
 lbm\_errmsg, 441  
 lbm\_errnum, 441  
 LBM\_ETIMEDOUT, 360  
 LBM\_EUMENOREG, 360  
 lbm\_event\_dispatch, 442  
 lbm\_event\_dispatch\_unblock, 442  
 lbm\_event\_queue\_attr\_create, 442  
 lbm\_event\_queue\_attr\_create\_default, 443  
 lbm\_event\_queue\_attr\_create\_from\_xml, 443  
 lbm\_event\_queue\_attr\_delete, 444  
 lbm\_event\_queue\_attr\_dump, 444  
 lbm\_event\_queue\_attr\_dup, 444  
 lbm\_event\_queue\_attr\_getopt, 445  
 lbm\_event\_queue\_attr\_option\_size, 445  
 lbm\_event\_queue\_attr\_set\_from\_xml, 445  
 lbm\_event\_queue\_attr\_setopt, 446  
 lbm\_event\_queue\_attr\_str\_getopt, 446  
 lbm\_event\_queue\_attr\_str\_setopt, 446  
 LBM\_EVENT\_QUEUE\_BLOCK, 360  
 lbm\_event\_queue\_cancel\_cb\_proc, 387  
 lbm\_event\_queue\_create, 447  
 LBM\_EVENT\_QUEUE\_DELAY\_WARNING, 360  
 lbm\_event\_queue\_delete, 447  
 lbm\_event\_queue\_dump, 448  
 LBM\_EVENT\_QUEUE\_ENQUEUE\_NOTIFICATION, 360  
 lbm\_event\_queue\_from\_rcv, 448  
 lbm\_event\_queue\_from\_src, 448  
 lbm\_event\_queue\_from\_wildcard\_rcv, 448  
 lbm\_event\_queue\_getopt, 449  
 lbm\_event\_queue\_monitor\_proc, 388  
 LBM\_EVENT\_QUEUE\_POLL, 361  
 lbm\_event\_queue\_reset\_stats, 449  
 lbm\_event\_queue\_retrieve\_stats, 449  
 lbm\_event\_queue\_setopt, 450  
 lbm\_event\_queue\_shutdown, 450  
 lbm\_event\_queue\_size, 450  
 LBM\_EVENT\_QUEUE\_SIZE\_WARNING, 361

lbm\_event\_queue\_stats\_t, 389  
lbm\_event\_queue\_str\_getopt, 451  
lbm\_event\_queue\_str\_setopt, 451  
LBM\_EWOULDBLOCK, 361  
lbm\_fd\_cb\_proc, 389  
LBM\_FD\_EVENT\_ACCEPT, 361  
LBM\_FD\_EVENT\_ALL, 361  
LBM\_FD\_EVENT\_CLOSE, 361  
LBM\_FD\_EVENT\_CONNECT,  
    361  
LBM\_FD\_EVENT\_EXCEPT, 361  
LBM\_FD\_EVENT\_READ, 361  
LBM\_FD\_EVENT\_WRITE, 361  
lbm\_flight\_size\_set\_inflight\_cb-  
    proc, 390  
lbm\_flight\_size\_set\_inflight\_ex-  
    cb\_proc, 390  
LBM\_FLIGHT\_SIZE\_TYPE\_ULB,  
    362  
LBM\_FLIGHT\_SIZE\_TYPE\_-  
    UME, 362  
LBM\_FLIGHT\_SIZE\_TYPE\_-  
    UMQ, 362  
lbm\_get\_jms\_msg\_id, 451  
lbm\_hf\_rcv\_create, 451  
lbm\_hf\_rcv\_delete, 452  
lbm\_hf\_rcv\_delete\_ex, 453  
lbm\_hf\_rcv\_from\_rcv, 453  
lbm\_hf\_rcv\_topic\_dump, 453  
lbm\_hf\_src\_create, 454  
lbm\_hf\_src\_send, 454  
lbm\_hf\_src\_send\_ex, 455  
lbm\_hf\_src\_send\_rcv\_reset, 456  
lbm\_hf\_src\_sendv, 457  
lbm\_hf\_src\_sendv\_ex, 458  
lbm\_hfx\_attr\_create, 459  
lbm\_hfx\_attr\_create\_default, 459  
lbm\_hfx\_attr\_create\_from\_xml, 459  
lbm\_hfx\_attr\_delete, 460  
lbm\_hfx\_attr\_dump, 460  
lbm\_hfx\_attr\_dup, 460  
lbm\_hfx\_attr\_getopt, 461  
lbm\_hfx\_attr\_option\_size, 461  
lbm\_hfx\_attr\_set\_from\_xml, 461  
lbm\_hfx\_attr\_setopt, 462  
lbm\_hfx\_attr\_str\_getopt, 462  
lbm\_hfx\_attr\_str\_setopt, 462  
lbm\_hfx\_create, 463  
lbm\_hfx\_delete, 463  
lbm\_hfx\_delete\_ex, 464  
lbm\_hfx\_dump, 464  
lbm\_hfx\_getopt, 465  
lbm\_hfx\_rcv\_create, 465  
lbm\_hfx\_rcv\_delete, 465  
lbm\_hfx\_rcv\_delete\_ex, 466  
lbm\_hfx\_rcv\_topic\_dump, 466  
lbm\_hfx\_setopt, 467  
lbm\_hfx\_str\_getopt, 467  
lbm\_hfx\_str\_setopt, 467  
lbm\_immediate\_msg\_cb\_proc, 390  
lbm\_iovec\_t, 391  
lbm\_ipv4\_address\_mask\_t, 391  
lbm\_is\_ume\_capable, 468  
lbm\_is\_umq\_capable, 468  
lbm\_license\_file, 468  
lbm\_license\_str, 468  
lbm\_log, 469  
LBM\_LOG\_ALERT, 362  
lbm\_log\_cb\_proc, 391  
LBM\_LOG\_CRIT, 362  
LBM\_LOG\_DEBUG, 362  
LBM\_LOG\_EMERG, 362  
LBM\_LOG\_ERR, 362  
LBM\_LOG\_INFO, 362  
LBM\_LOG\_NOTICE, 362  
LBM\_LOG\_WARNING, 363  
lbm\_logf, 469  
lbm\_mim\_unrecloss\_func\_t, 392  
lbm\_mim\_unrecloss\_function\_cb,  
    392  
LBM\_MSG\_BOS, 363  
lbm\_msg\_channel\_info\_t, 392  
LBM\_MSG\_COMPLETE\_BATCH,  
    363  
LBM\_MSG\_DATA, 363  
lbm\_msg\_delete, 469  
LBM\_MSG\_END\_BATCH, 363  
LBM\_MSG\_EOS, 363  
lbm\_msg\_extract\_ume\_ack, 470  
LBM\_MSG\_FLAG\_DELIVERY\_-  
    LATENCY, 363

**LBM\_MSG\_FLAG\_END\_BATCH,**  
 363  
**LBM\_MSG\_FLAG\_HF\_32,** 363  
**LBM\_MSG\_FLAG\_HF\_64,** 364  
**LBM\_MSG\_FLAG\_HF\_-**  
 DUPLICATE, 364  
**LBM\_MSG\_FLAG\_HF\_-**  
 OPTIONAL, 364  
**LBM\_MSG\_FLAG\_HF\_PASS\_-**  
 THROUGH, 364  
**LBM\_MSG\_FLAG\_IMMEDIATE,**  
 364  
**LBM\_MSG\_FLAG\_-**  
 NUMBERED\_CHANNEL,  
 364  
**LBM\_MSG\_FLAG\_OTR,** 364  
**LBM\_MSG\_FLAG\_-**  
 RETRANSMIT, 364  
**LBM\_MSG\_FLAG\_START\_-**  
 BATCH, 364  
**LBM\_MSG\_FLAG\_TOPICLESS,**  
 364  
**LBM\_MSG\_FLAG\_UME\_-**  
 RETRANSMIT, 365  
**LBM\_MSG\_FLAG\_UMQ\_-**  
 REASSIGNED, 365  
**LBM\_MSG\_FLAG\_UMQ\_-**  
 RESUBMITTED, 365  
**LBM\_MSG\_FLUSH,** 365  
**lbm\_msg\_fragment\_info\_t,** 392  
**lbm\_msg\_gateway\_info\_t,** 392  
**LBM\_MSG\_HF\_RESET,** 365  
**LBM\_MSG iov\_GATHER,** 365  
**lbm\_msg\_is\_fragment,** 470  
**LBM\_MSG\_NO\_SOURCE\_-**  
 NOTIFICATION, 365  
**lbm\_msg\_properties\_clear,** 470  
**lbm\_msg\_properties\_create,** 470  
**lbm\_msg\_properties\_delete,** 471  
**lbm\_msg\_properties\_get,** 471  
**lbm\_msg\_properties\_iter\_create,**  
 472  
**lbm\_msg\_properties\_iter\_delete,**  
 472  
**lbm\_msg\_properties\_iter\_first,** 473  
**lbm\_msg\_properties\_iter\_next,** 473  
**LBM\_MSG\_PROPERTIES\_-**  
 MAX\_NAMELEN, 365  
**lbm\_msg\_properties\_set,** 474  
**LBM\_MSG\_PROPERTY\_-**  
 BOOLEAN, 366  
**LBM\_MSG\_PROPERTY\_BYTEx**,  
 366  
**LBM\_MSG\_PROPERTY\_-**  
 DOUBLE, 366  
**LBM\_MSG\_PROPERTY\_FLOAT,**  
 366  
**LBM\_MSG\_PROPERTY\_INT,** 366  
**LBM\_MSG\_PROPERTY\_LONG,**  
 366  
**LBM\_MSG\_PROPERTY\_NONE,**  
 366  
**LBM\_MSG\_PROPERTY\_SHORT,**  
 366  
**LBM\_MSG\_PROPERTY\_STRING,**  
 366  
**LBM\_MSG\_REQUEST,** 366  
**LBM\_MSG\_RESPONSE,** 367  
**lbm\_msg\_retain,** 474  
**lbm\_msg\_retrieve\_fragment\_info,**  
 475  
**lbm\_msg\_retrieve\_gateway\_info,**  
 475  
**lbm\_msg\_retrievemsgid,** 475  
**lbm\_msg\_retrieve\_umq\_index,** 475  
**LBM\_MSG\_START\_BATCH,** 367  
**lbm\_msg\_ume\_can\_send\_explicit\_-**  
 ack, 476  
**LBM\_MSG\_UME\_-**  
 DEREGISTRATION\_-  
 COMPLETE\_EX, 367  
**lbm\_ume\_deregistration\_ex\_t,**  
 393  
**LBM\_MSG\_UME\_-**  
 DEREGISTRATION\_-  
 SUCCESS\_EX, 367  
**LBM\_MSG\_UME\_-**  
 DEREGISTRATION\_-  
 SUCCESS\_EX\_FLAG\_RPP,  
 367  
**LBM\_MSG\_UME\_-**  
 REGISTRATION\_CHANGE,

367  
LBM\_MSG\_UME\_-  
REGISTRATION\_-  
COMPLETE\_EX, 367  
LBM\_MSG\_UME\_-  
REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_-  
QUORUM, 367  
LBM\_MSG\_UME\_-  
REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_-  
RXREQMAX, 368  
lbm\_msg\_ume\_registration\_-  
complete\_ex\_t, 393  
LBM\_MSG\_UME\_-  
REGISTRATION\_ERROR,  
368  
lbm\_msg\_ume\_registration\_ex\_t,  
393  
LBM\_MSG\_UME\_-  
REGISTRATION\_SUCCESS,  
368  
LBM\_MSG\_UME\_-  
REGISTRATION\_-  
SUCCESS\_EX, 368  
LBM\_MSG\_UME\_-  
REGISTRATION\_-  
SUCCESS\_EX\_FLAG\_-  
NOCACHE, 368  
LBM\_MSG\_UME\_-  
REGISTRATION\_-  
SUCCESS\_EX\_FLAG\_OLD,  
368  
LBM\_MSG\_UME\_-  
REGISTRATION\_-  
SUCCESS\_EX\_FLAG\_RPP,  
368  
lbm\_msg\_ume\_registration\_t, 393  
lbm\_msg\_ume\_send\_explicit\_ack,  
476  
LBM\_MSG\_UMQ\_-  
DEREGISTRATION\_-  
COMPLETE\_EX, 368  
LBM\_MSG\_UMQ\_-  
DEREGISTRATION\_-  
COMPLETE\_EX\_FLAG\_-  
ULB, 369  
lbm\_msg\_umq\_deregistration\_-  
complete\_ex\_t, 393  
LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNED\_EX, 369  
LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNED\_EX\_FLAG\_-  
REQUESTED, 369  
LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNED\_EX\_FLAG\_ULB,  
369  
lbm\_msg\_umq\_index\_assigned\_-  
ex\_t, 393  
LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNMENT\_-  
ELIGIBILITY\_ERROR,  
369  
LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNMENT\_-  
ELIGIBILITY\_START\_-  
COMPLETE\_EX, 369  
LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNMENT\_-  
ELIGIBILITY\_START\_-  
COMPLETE\_EX\_FLAG\_-  
ULB, 369  
lbm\_msg\_umq\_index\_assignment\_-  
eligibility\_start\_complete\_ex\_t,  
393  
LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNMENT\_-  
ELIGIBILITY\_STOP\_-  
COMPLETE\_EX, 369  
LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNMENT\_-  
ELIGIBILITY\_STOP\_-  
COMPLETE\_EX\_FLAG\_-  
ULB, 370  
lbm\_msg\_umq\_index\_assignment\_-  
eligibility\_stop\_complete\_ex\_t,  
394  
LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNMENT\_ERROR,  
370  
LBM\_MSG\_UMQ\_INDEX\_-  
RELEASED\_EX, 370

**LBM\_MSG\_UMQ\_INDEX\_-RELEASED\_EX\_FLAG\_-ULB,** 370  
 lbm\_msg\_umq\_index\_released\_ex\_t, 394  
 lbm\_msg\_umq\_reassign, 476  
**LBM\_MSG\_UMQ\_REASSIGN\_-FLAG\_DISCARD,** 370  
**LBM\_MSG\_UMQ\_-REGISTRATION\_-COMPLETE\_EX,** 370  
**LBM\_MSG\_UMQ\_-REGISTRATION\_-COMPLETE\_EX\_FLAG\_-QUORUM,** 370  
**LBM\_MSG\_UMQ\_-REGISTRATION\_-COMPLETE\_EX\_FLAG\_-ULB,** 370  
 lbm\_msg\_umq\_registration\_complete\_ex\_t, 394  
**LBM\_MSG\_UMQ\_-REGISTRATION\_ERROR,** 371  
**LBM\_MSG\_-UNRECOVERABLE\_LOSS,** 371  
**LBM\_MSG\_-UNRECOVERABLE\_LOSS\_-BURST,** 371  
 lbm\_multicast\_immediate\_message, 477  
 lbm\_multicast\_immediate\_request, 477  
 lbm\_queue\_immediate\_message, 478  
**LBM\_RCV\_BLOCK,** 371  
**LBM\_RCV\_BLOCK\_TEMP,** 371  
 lbm\_rcv\_cb\_proc, 394  
 lbm\_rcv\_create, 478  
 lbm\_rcv\_delete, 479  
 lbm\_rcv\_delete\_ex, 480  
 lbm\_rcv\_from\_hf\_rcv, 480  
 lbm\_rcv\_from\_hfx\_rcv, 480  
 lbm\_rcv\_getopt, 480  
 lbm\_rcv\_msg\_source\_clientd, 481  
**LBM\_RCV\_NONBLOCK,** 371  
 lbm\_rcv\_reset\_all\_transport\_stats, 481  
 lbm\_rcv\_reset\_transport\_stats, 481  
 lbm\_rcv\_retrieve\_all\_transport\_stats, 482  
 lbm\_rcv\_retrieve\_transport\_stats, 482  
 lbm\_rcv\_setopt, 482  
 lbm\_rcv\_src\_notification\_create\_function\_cb, 395  
 lbm\_rcv\_src\_notification\_delete\_function\_cb, 395  
 lbm\_rcv\_src\_notification\_func\_t, 396  
 lbm\_rcv\_str\_getopt, 483  
 lbm\_rcv\_str\_setopt, 483  
 lbm\_rcv\_subscribe\_channel, 484  
 lbm\_rcv\_topic\_attr\_create, 484  
 lbm\_rcv\_topic\_attr\_create\_default, 485  
 lbm\_rcv\_topic\_attr\_create\_from\_xml, 485  
 lbm\_rcv\_topic\_attr\_delete, 485  
 lbm\_rcv\_topic\_attr\_dump, 486  
 lbm\_rcv\_topic\_attr\_dup, 486  
 lbm\_rcv\_topic\_attr\_getopt, 486  
 lbm\_rcv\_topic\_attr\_option\_size, 487  
 lbm\_rcv\_topic\_attr\_set\_from\_xml, 487  
 lbm\_rcv\_topic\_attr\_setopt, 487  
 lbm\_rcv\_topic\_attr\_str\_getopt, 488  
 lbm\_rcv\_topic\_attr\_str\_setopt, 488  
 lbm\_rcv\_topic\_dump, 489  
 lbm\_rcv\_topic\_lookup, 489  
 lbm\_rcv\_transport\_stats\_daemon\_t, 396  
 lbm\_rcv\_transport\_stats\_t, 396  
 lbm\_rcv\_ume\_deregister, 490  
 lbm\_rcv\_umq\_deregister, 490  
 lbm\_rcv\_umq\_index\_release, 490  
 lbm\_rcv\_umq\_index\_reserve, 490  
 lbm\_rcv\_umq\_index\_start\_assignment, 491  
 lbm\_rcv\_umq\_index\_stop\_assignment, 491

lbtm\_recv\_umq\_queue\_msg\_list, 492  
lbtm\_recv\_umq\_queue\_msg\_retrieve,  
    492  
lbtm\_recv\_unsubscribe\_channel, 493  
lbtm\_recv\_unsubscribe\_channel\_ex,  
    493  
lbtm\_register\_fd, 494  
lbtm\_request\_cb\_proc, 396  
lbtm\_request\_delete, 494  
lbtm\_request\_delete\_ex, 495  
lbtm\_response\_delete, 495  
lbtm\_schedule\_timer, 495  
lbtm\_schedule\_timer\_recurring, 496  
lbtm\_send\_request, 497  
lbtm\_send\_request\_ex, 497  
lbtm\_send\_response, 498  
lbtm\_serialize\_response, 499  
lbtm\_serialized\_response\_delete, 499  
lbtm\_set\_lbtrm\_loss\_rate, 499  
lbtm\_set\_lbtrm\_src\_loss\_rate, 500  
lbtm\_set\_lbtru\_loss\_rate, 500  
lbtm\_set\_lbtru\_src\_loss\_rate, 500  
lbtm\_set\_umm\_info, 500  
LBM\_SRC\_BLOCK, 371  
LBM\_SRC\_BLOCK\_TEMP, 371  
lbtm\_src\_cb\_proc, 396  
lbtm\_src\_channel\_create, 501  
lbtm\_src\_channel\_delete, 501  
lbtm\_src\_cost\_function\_cb, 398  
LBM\_SRC\_COST\_FUNCTION\_-  
    REJECT, 371  
lbtm\_src\_create, 501  
lbtm\_src\_delete, 502  
lbtm\_src\_delete\_ex, 502  
LBM\_SRC\_EVENT\_CONNECT,  
    372  
LBM\_SRC\_EVENT\_DAEMON\_-  
    CONFIRM, 372  
LBM\_SRC\_EVENT\_-  
    DISCONNECT, 372  
LBM\_SRC\_EVENT\_FLIGHT\_-  
    SIZE\_NOTIFICATION, 372  
LBM\_SRC\_EVENT\_FLIGHT\_-  
    SIZE\_NOTIFICATION\_-  
    STATE\_OVER, 372  
LBM\_SRC\_EVENT\_FLIGHT\_-  
    SIZE\_NOTIFICATION\_-  
    STATE\_UNDER, 372  
lbtm\_src\_event\_flight\_size\_-  
    notification\_t, 399  
LBM\_SRC\_EVENT\_FLIGHT\_-  
    SIZE\_NOTIFICATION\_-  
    TYPE\_ULB, 373  
LBM\_SRC\_EVENT\_FLIGHT\_-  
    SIZE\_NOTIFICATION\_-  
    TYPE\_UME, 373  
LBM\_SRC\_EVENT\_FLIGHT\_-  
    SIZE\_NOTIFICATION\_-  
    TYPE\_UMQ, 373  
LBM\_SRC\_EVENT\_-  
    SEQUENCE\_NUMBER\_-  
    INFO, 373  
lbtm\_src\_event\_sequence\_number\_-  
    info\_t, 399  
lbtm\_src\_event\_ume\_ack\_ex\_info\_t,  
    399  
lbtm\_src\_event\_ume\_ack\_info\_t,  
    399  
LBM\_SRC\_EVENT\_-  
    UME\_DELIVERY\_-  
    CONFIRMATION, 373  
LBM\_SRC\_EVENT\_-  
    UME\_DELIVERY\_-  
    CONFIRMATION\_EX, 373  
LBM\_SRC\_EVENT\_-  
    UME\_DELIVERY\_-  
    CONFIRMATION\_EX\_-  
    FLAG\_EXACK, 373  
LBM\_SRC\_EVENT\_-  
    UME\_DELIVERY\_-  
    CONFIRMATION\_EX\_-  
    FLAG\_OOD, 374  
LBM\_SRC\_EVENT\_-  
    UME\_DELIVERY\_-  
    CONFIRMATION\_EX\_-  
    FLAG\_UNIQUEACKS, 374  
LBM\_SRC\_EVENT\_-  
    UME\_DELIVERY\_-  
    CONFIRMATION\_EX\_-  
    FLAG\_UREGID, 374

LBM\_SRC\_EVENT\_-
   
     UME\_DELIVERY\_-
   
     CONFIRMATION\_EX\_-
   
     FLAG\_WHOLE\_MESSAGE\_-
   
     CONFIRMED, 374

LBM\_SRC\_EVENT\_UME\_-
   
     Deregistration\_-
   
     COMPLETE\_EX, 374

lbm\_src\_event\_ume\_-
   
     deregistration\_ex\_t, 399

LBM\_SRC\_EVENT\_UME\_-
   
     Deregistration\_-
   
     SUCCESS\_EX, 374

LBM\_SRC\_EVENT\_UME\_-
   
     MESSAGE\_RECLAIMED,
   
     374

LBM\_SRC\_EVENT\_UME\_-
   
     MESSAGE\_RECLAIMED -
   
     EX, 375

LBM\_SRC\_EVENT\_UME\_-
   
     MESSAGE\_RECLAIMED\_-
   
     EX\_FLAG\_FORCED, 375

LBM\_SRC\_EVENT\_UME\_-
   
     MESSAGE\_STABLE, 375

LBM\_SRC\_EVENT\_UME\_-
   
     MESSAGE\_STABLE\_EX,
   
     375

LBM\_SRC\_EVENT\_UME\_-
   
     MESSAGE\_STABLE\_EX\_-
   
     FLAG\_INTEGRGROUP\_-
   
     STABLE, 375

LBM\_SRC\_EVENT\_UME\_-
   
     MESSAGE\_STABLE\_EX\_-
   
     FLAG\_INTRAGROUP\_-
   
     STABLE, 375

LBM\_SRC\_EVENT\_UME\_-
   
     MESSAGE\_STABLE\_EX\_-
   
     FLAG\_STORE, 375

LBM\_SRC\_EVENT\_UME\_-
   
     MESSAGE\_STABLE\_EX\_-
   
     FLAG\_USER, 376

LBM\_SRC\_EVENT\_UME\_-
   
     MESSAGE\_STABLE\_EX\_-

FLAG\_WHOLE\_MESSAGE\_-
   
     STABLE, 376

LBM\_SRC\_EVENT\_UME\_-
   
     REGISTRATION\_-
   
     COMPLETE\_EX, 376

LBM\_SRC\_EVENT\_UME\_-
   
     REGISTRATION\_-
   
     COMPLETE\_EX\_FLAG\_-
   
     QUORUM, 376

lbm\_src\_event\_ume\_registration\_-
   
     complete\_ex\_t, 399

LBM\_SRC\_EVENT\_UME\_-
   
     REGISTRATION\_ERROR,
   
     376

lbm\_src\_event\_ume\_registration\_-
   
     ex\_t, 400

LBM\_SRC\_EVENT\_UME\_-
   
     REGISTRATION\_SUCCESS,
   
     376

LBM\_SRC\_EVENT\_UME\_-
   
     REGISTRATION\_-
   
     SUCCESS\_EX, 376

LBM\_SRC\_EVENT\_UME\_-
   
     REGISTRATION\_-
   
     SUCCESS\_EX\_FLAG\_-
   
     NOACKS, 376

LBM\_SRC\_EVENT\_UME\_-
   
     REGISTRATION\_-
   
     SUCCESS\_EX\_FLAG\_OLD,
   
     377

LBM\_SRC\_EVENT\_UME\_-
   
     REGISTRATION\_-
   
     SUCCESS\_EX\_FLAG\_RPP,
   
     377

lbm\_src\_event\_ume\_registration\_t,
   
     400

LBM\_SRC\_EVENT\_UME\_-
   
     STORE\_UNRESPONSIVE,
   
     377

LBM\_SRC\_EVENT\_UMQ\_-
   
     MESSAGE\_ID\_INFO, 377

lbm\_src\_event\_umq\_message\_id\_-
   
     info\_t, 400

LBM\_SRC\_EVENT\_UMQ\_-
   
     MESSAGE\_STABLE\_EX,
   
     377

- LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_INTEGRGROUP\_-  
STABLE, 377
- LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_INTRAGROUP\_-  
STABLE, 377
- LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_STABLE, 378
- LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_USER, 378
- LBM\_SRC\_EVENT\_UMQ\_-  
REGISTRATION\_-  
COMPLETE\_EX, 378
- LBM\_SRC\_EVENT\_UMQ\_-  
REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_-  
QUORUM, 378
- lbt\_src\_event\_umq\_registration\_-  
complete\_ex\_t, 400
- LBM\_SRC\_EVENT\_UMQ\_-  
REGISTRATION\_ERROR,  
378
- lbt\_src\_event\_umq\_stability\_ack\_-  
info\_ex\_t, 400
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_ASSIGNED\_EX,  
378
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_COMPLETE\_EX,  
378
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_CONSUMED\_-  
EX, 379
- lbt\_src\_event\_umq\_ulb\_message\_-  
info\_ex\_t, 400
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_REASSIGNED\_-  
EX, 379
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_REASSIGNED\_-  
EX\_FLAG\_EXPLICIT, 379
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_TIMEOUT\_EX,  
379
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_TIMEOUT\_EX\_-  
FLAG\_DISCARD, 379
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_TIMEOUT\_EX\_-  
FLAG\_EXPLICIT, 379
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_TIMEOUT\_EX\_-  
FLAG\_MAX\_REASSIGNS,  
379
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_TIMEOUT\_EX\_-  
FLAG\_TOTAL\_LIFETIME\_-  
EXPIRED, 380
- LBM\_SRC\_EVENT\_UMQ\_-  
ULB\_RECEIVER\_-  
Deregistration\_EX,  
380
- lbt\_src\_event\_umq\_ulb\_receiver\_-  
info\_ex\_t, 400
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
RECEIVER\_READY\_EX,  
380
- LBM\_SRC\_EVENT\_UMQ\_-  
ULB\_RECEIVER\_-  
REGISTRATION\_EX, 380
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
RECEIVER\_TIMEOUT\_EX,  
380
- LBM\_SRC\_EVENT\_WAKEUP,  
380
- LBM\_SRC\_EVENT\_WAKEUP\_-  
FLAG\_MIM, 380
- LBM\_SRC\_EVENT\_WAKEUP\_-  
FLAG\_NORMAL, 381
- LBM\_SRC\_EVENT\_WAKEUP\_-  
FLAG\_REQUEST, 381
- LBM\_SRC\_EVENT\_WAKEUP\_-  
FLAG\_RESPONSE, 381
- LBM\_SRC\_EVENT\_WAKEUP\_-  
FLAG\_UIM, 381
- lbt\_src\_event\_wakeup\_t, 401
- lbt\_src\_flush, 503

lbm\_src\_get\_inflight, 503  
 lbm\_src\_get\_inflight\_ex, 504  
 lbm\_src\_getopt, 504  
 LBM\_SRC\_NONBLOCK, 381  
 lbm\_src\_notify\_func\_t, 401  
 lbm\_src\_notify\_function\_cb, 401  
 lbm\_src\_reset\_transport\_stats, 505  
 lbm\_src\_retrieve\_transport\_stats,  
     505  
 lbm\_src\_send, 505  
 lbm\_src\_send\_ex, 506  
 LBM\_SRC\_SEND\_EX\_FLAG\_-  
     APPHDR\_CHAIN, 381  
 LBM\_SRC\_SEND\_EX\_FLAG\_-  
     CHANNEL, 381  
 LBM\_SRC\_SEND\_EX\_FLAG\_-  
     HF\_32, 381  
 LBM\_SRC\_SEND\_EX\_FLAG\_-  
     HF\_64, 381  
 LBM\_SRC\_SEND\_EX\_FLAG\_-  
     HF\_OPTIONAL, 382  
 LBM\_SRC\_SEND\_EX\_FLAG\_-  
     PROPERTIES, 382  
 LBM\_SRC\_SEND\_EX\_FLAG\_-  
     SEQUENCE\_NUMBER\_-  
     INFO, 382  
 LBM\_SRC\_SEND\_EX\_FLAG\_-  
     SEQUENCE\_NUMBER\_-  
     INFO\_FRAGONLY, 382  
 LBM\_SRC\_SEND\_EX\_FLAG\_-  
     UME\_CLIENTD, 382  
 LBM\_SRC\_SEND\_EX\_FLAG\_-  
     UMQ\_CLIENTD, 382  
 LBM\_SRC\_SEND\_EX\_FLAG\_-  
     UMQ\_INDEX, 382  
 LBM\_SRC\_SEND\_EX\_FLAG\_-  
     UMQ\_MESSAGE\_ID\_INFO,  
     382  
 LBM\_SRC\_SEND\_EX\_FLAG\_-  
     UMQ\_TOTAL\_LIFETIME,  
     382  
 lbm\_src\_send\_ex\_info\_t, 401  
 lbm\_src\_sendv, 507  
 lbm\_src\_sendv\_ex, 508  
 lbm\_src\_setopt, 509  
 lbm\_src\_str\_getopt, 509  
 lbm\_src\_str\_setopt, 509  
 lbm\_src\_topic\_alloc, 510  
 lbm\_src\_topic\_attr\_create, 510  
 lbm\_src\_topic\_attr\_create\_default,  
     511  
 lbm\_src\_topic\_attr\_create\_from\_-  
     xml, 511  
 lbm\_src\_topic\_attr\_delete, 512  
 lbm\_src\_topic\_attr\_dump, 512  
 lbm\_src\_topic\_attr\_dup, 512  
 lbm\_src\_topic\_attr\_getopt, 512  
 lbm\_src\_topic\_attr\_option\_size, 513  
 lbm\_src\_topic\_attr\_set\_from\_xml,  
     513  
 lbm\_src\_topic\_attr\_setopt, 513  
 lbm\_src\_topic\_attr\_str\_getopt, 514  
 lbm\_src\_topic\_attr\_str\_setopt, 514  
 lbm\_src\_topic\_dump, 515  
 lbm\_src\_transport\_stats\_daemon\_t,  
     401  
 lbm\_src\_transport\_stats\_t, 402  
 lbm\_src\_ume\_deregister, 515  
 lbm\_str\_hash\_func\_ex\_t, 402  
 lbm\_str\_hash\_function\_cb, 402  
 lbm\_str\_hash\_function\_cb\_ex, 402  
 lbm\_timer\_cb\_proc, 403  
 lbm\_timeval\_t, 403  
 lbm\_topic\_from\_src, 515  
 LBM\_TOPIC\_RES\_REQUEST\_-  
     ADVERTISEMENT, 383  
 LBM\_TOPIC\_RES\_REQUEST\_-  
     QUERY, 383  
 LBM\_TOPIC\_RES\_REQUEST\_-  
     RESERVED1, 383  
 LBM\_TOPIC\_RES\_REQUEST\_-  
     WILDCARD\_QUERY, 383  
 lbm\_transport\_source\_format, 515  
 lbm\_transport\_source\_info\_t, 403  
 lbm\_transport\_source\_parse, 516  
 LBM\_TRANSPORT\_STAT\_-  
     DAEMON, 383  
 LBM\_TRANSPORT\_STAT\_-  
     LBTIPC, 383  
 LBM\_TRANSPORT\_STAT\_-  
     LBTRDMA, 383

LBM\_TRANSPORT\_STAT\_-  
LBTRM, 383  
LBM\_TRANSPORT\_STAT\_-  
LBTRU, 384  
LBM\_TRANSPORT\_STAT\_TCP,  
384  
LBM\_TRANSPORT\_TYPE\_-  
LBTPC, 384  
LBM\_TRANSPORT\_TYPE\_-  
LBTRDMA, 384  
LBM\_TRANSPORT\_TYPE\_-  
LBTRM, 384  
LBM\_TRANSPORT\_TYPE\_-  
LBTRU, 384  
LBM\_TRANSPORT\_TYPE\_TCP,  
384  
lbm\_icast\_resolver\_entry\_t, 404  
lbm\_ume\_ack\_delete, 516  
lbm\_ume\_ack\_send\_explicit\_ack,  
516  
lbm\_ume\_ctx\_recv\_ctx\_-  
notification\_create\_function\_-  
cb, 404  
lbm\_ume\_ctx\_recv\_ctx\_-  
notification\_delete\_function\_-  
cb, 404  
lbm\_ume\_ctx\_recv\_ctx\_-  
notification\_func\_t, 405  
lbm\_ume\_rcv\_recovery\_info\_ex\_-  
func\_info\_t, 405  
lbm\_ume\_rcv\_recovery\_info\_ex\_-  
func\_t, 405  
lbm\_ume\_rcv\_recovery\_info\_ex\_-  
function\_cb, 405  
lbm\_ume\_rcv\_regid\_ex\_func\_info\_-  
t, 406  
lbm\_ume\_rcv\_regid\_ex\_func\_t, 406  
lbm\_ume\_rcv\_regid\_ex\_function\_-  
cb, 406  
lbm\_ume\_rcv\_regid\_func\_t, 407  
lbm\_ume\_rcv\_regid\_function\_cb,  
407  
lbm\_ume\_src\_force\_reclaim\_func\_-  
t, 407  
lbm\_ume\_src\_force\_reclaim\_-  
function\_cb, 407  
lbm\_ume\_src\_msg\_stable, 517  
lbm\_ume\_store\_entry\_t, 408  
lbm\_ume\_store\_group\_entry\_t, 408  
lbm\_ume\_store\_name\_entry\_t, 408  
LBM\_UMM\_INFO\_FLAGS\_-  
USE\_SSL, 384  
lbm\_umq\_ctx\_msg\_stable, 517  
LBM\_UMQ\_INDEX\_FLAG\_-  
NUMERIC, 384  
lbm\_umq\_index\_info\_t, 408  
lbm\_umq\_msg\_selector\_create, 517  
lbm\_umq\_msg\_selector\_delete, 518  
lbm\_umq\_msg\_total\_lifetime\_-  
info\_t, 408  
lbm\_umqmsgid\_t, 409  
lbm\_umq\_queue\_entry\_t, 409  
lbm\_umqregid\_t, 409  
lbm\_umq\_ulb\_application\_set\_-  
attr\_t, 409  
lbm\_umq\_ulb\_receiver\_type\_attr\_t,  
409  
lbm\_umq\_ulb\_receiver\_type\_-  
entry\_t, 409  
lbm\_unicast\_immediate\_message,  
518  
lbm\_unicast\_immediate\_request,  
518  
lbm\_version, 519  
lbm\_wildcard\_rcv\_attr\_create, 519  
lbm\_wildcard\_rev\_attr\_create\_-  
default, 520  
lbm\_wildcard\_rcv\_attr\_create\_-  
from\_xml, 520  
lbm\_wildcard\_rev\_attr\_delete, 521  
lbm\_wildcard\_rev\_attr\_dump, 521  
lbm\_wildcard\_rev\_attr\_dup, 521  
lbm\_wildcard\_rev\_attr\_getopt, 522  
lbm\_wildcard\_rev\_attr\_option\_size,  
522  
lbm\_wildcard\_rev\_attr\_set\_from\_-  
xml, 522  
lbm\_wildcard\_rcv\_attr\_setopt, 523  
lbm\_wildcard\_rev\_attr\_str\_getopt,  
523  
lbm\_wildcard\_rev\_attr\_str\_setopt,  
523

lbm\_wildcard\_rcv\_compare\_func\_t,  
     **409**  
 lbm\_wildcard\_rcv\_compare\_-  
     function\_cb, **409**  
 lbm\_wildcard\_rcv\_create, **524**  
 lbm\_wildcard\_rcv\_create\_func\_t,  
     **410**  
 lbm\_wildcard\_rcv\_create\_-  
     function\_cb, **410**  
 lbm\_wildcard\_rcv\_delete, **525**  
 lbm\_wildcard\_rcv\_delete\_ex, **525**  
 lbm\_wildcard\_rcv\_delete\_func\_t,  
     **411**  
 lbm\_wildcard\_rcv\_delete\_-  
     function\_cb, **411**  
 lbm\_wildcard\_rcv\_dump, **525**  
 lbm\_wildcard\_rcv\_getopt, **526**  
 LBM\_WILDCARD\_RCV\_-  
     PATTERN\_TYPE\_APP\_CB,  
     **385**  
 LBM\_WILDCARD\_RCV\_-  
     PATTERN\_TYPE\_PCRE,  
     **385**  
 LBM\_WILDCARD\_RCV\_-  
     PATTERN\_TYPE\_REGEX,  
     **385**  
 lbm\_wildcard\_rcv\_setopt, **526**  
 lbm\_wildcard\_rcv\_str\_getopt, **526**  
 lbm\_wildcard\_rcv\_str\_setopt, **527**  
 lbm\_wildcard\_rcv\_subscribe\_-  
     channel, **527**  
 lbm\_wildcard\_rcv\_umq\_deregister,  
     **528**  
 lbm\_wildcard\_rcv\_umq\_index\_-  
     release, **528**  
 lbm\_wildcard\_rcv\_umq\_index\_-  
     start\_assignment, **528**  
 lbm\_wildcard\_rcv\_umq\_index\_-  
     stop\_assignment, **529**  
 lbm\_wildcard\_rcv\_unsubscribe\_-  
     channel, **529**  
 lbm\_wildcard\_rcv\_unsubscribe\_-  
     channel\_ex, **529**  
 lbm\_win32\_static\_thread\_attach,  
     **530**  
 lbm\_win32\_static\_thread\_detach,  
     **530**  
 lbm\_wrcv\_ume\_deregister, **530**  
 ume\_liveness\_receiving\_context\_t,  
     **411**  
 lbm\_apphdr\_chain\_append\_elem  
     lbm.h, **412**  
 lbm\_apphdr\_chain\_create  
     lbm.h, **412**  
 lbm\_apphdr\_chain\_delete  
     lbm.h, **412**  
 lbm\_apphdr\_chain\_elem\_t\_stct, **113**  
     data, **114**  
     len, **113**  
     subtype, **113**  
     type, **113**  
 lbm\_apphdr\_chain\_iter\_create  
     lbm.h, **412**  
 lbm\_apphdr\_chain\_iter\_create\_from\_-  
     msg  
     lbm.h, **413**  
 lbm\_apphdr\_chain\_iter\_current  
     lbm.h, **413**  
 lbm\_apphdr\_chain\_iter\_delete  
     lbm.h, **413**  
 lbm\_apphdr\_chain\_iter\_done  
     lbm.h, **413**  
 lbm\_apphdr\_chain\_iter\_first  
     lbm.h, **414**  
 lbm\_apphdr\_chain\_iter\_next  
     lbm.h, **414**  
 LBM\_ASYNC\_OP\_INFO\_FLAG\_-  
     FIRST  
     lbm.h, **356**  
 LBM\_ASYNC\_OP\_INFO\_FLAG\_-  
     INLINE  
     lbm.h, **356**  
 LBM\_ASYNC\_OP\_INFO\_FLAG\_LAST  
     lbm.h, **356**  
 LBM\_ASYNC\_OP\_INFO\_FLAG\_-  
     ONLY  
     lbm.h, **356**  
 LBM\_ASYNC\_OP\_INVALID\_-  
     HANDLE  
     lbm.h, **356**

LBM\_ASYNC\_OP\_STATUS\_-  
  CANCELED  
    lbm.h, 356

LBM\_ASYNC\_OP\_STATUS\_-  
  COMPLETE  
    lbm.h, 356

LBM\_ASYNC\_OP\_STATUS\_ERROR  
    lbm.h, 356

LBM\_ASYNC\_OP\_STATUS\_IN\_-  
  PROGRESS  
    lbm.h, 356

LBM\_ASYNC\_OP\_TYPE\_CTX\_-  
  UMQ\_QUEUE\_TOPIC\_LIST  
    lbm.h, 357

LBM\_ASYNC\_OP\_TYPE\_RCV\_-  
  UMQ\_QUEUE\_MSG\_LIST  
    lbm.h, 357

LBM\_ASYNC\_OP\_TYPE\_RCV\_-  
  UMQ\_QUEUE\_MSG\_-  
  RETRIEVE  
    lbm.h, 357

lbm\_async\_operation\_cancel  
  lbm.h, 414

LBM\_ASYNC\_OPERATION\_-  
  CANCEL\_FLAG\_-  
  NONBLOCK  
    lbm.h, 357

lbm\_async\_operation\_func\_t, 115  
  clientd, 115  
  evq, 115  
  flags, 115  
  func, 115

lbm\_async\_operation\_function\_cb  
  lbm.h, 385

lbm\_async\_operation\_info\_t, 116  
  flags, 117  
  handle, 117  
  info, 117  
  status, 117  
  type, 117

lbm\_async\_operation\_status  
  lbm.h, 415

LBM\_ASYNC\_OPERATION\_-  
  STATUS\_FLAG\_NONBLOCK  
    lbm.h, 357

lbm\_auth\_set\_credentials

    lbm.h, 416

lbm\_authstorage\_addtpnam  
  lbm.h, 416

lbm\_authstorage\_checkpermission  
  lbm.h, 417

lbm\_authstorage\_close\_storage\_xml  
  lbm.h, 417

lbm\_authstorage\_deltpnam  
  lbm.h, 417

lbm\_authstorage\_load\_roletable  
  lbm.h, 417

lbm\_authstorage\_open\_storage\_xml  
  lbm.h, 418

lbm\_authstorage\_print\_roletable  
  lbm.h, 418

lbm\_authstorage\_roletable\_add\_role\_-  
  action  
    lbm.h, 418

lbm\_authstorage\_unload\_roletable  
  lbm.h, 419

lbm\_authstorage\_user\_add\_role  
  lbm.h, 419

lbm\_authstorage\_user\_del\_role  
  lbm.h, 419

lbm\_cancel\_fd  
  lbm.h, 420

lbm\_cancel\_fd\_ex  
  lbm.h, 420

lbm\_cancel\_timer  
  lbm.h, 421

lbm\_cancel\_timer\_ex  
  lbm.h, 422

LBM\_CHAIN\_ELEM\_APPHDR  
  lbm.h, 357

LBM\_CHAIN\_ELEM\_CHANNEL\_-  
  NUMBER  
    lbm.h, 357

LBM\_CHAIN\_ELEM\_GW\_INFO  
  lbm.h, 357

LBM\_CHAIN\_ELEM\_HF\_SQN  
  lbm.h, 358

LBM\_CHAIN\_ELEM\_PROPERTIES\_-  
  LENGTH  
    lbm.h, 358

LBM\_CHAIN\_ELEM\_USER\_DATA  
  lbm.h, 358

lbm\_config  
     lbm.h, 422  
 lbm\_config\_xml\_file  
     lbm.h, 422  
 lbm\_config\_xml\_string  
     lbm.h, 423  
 lbm\_context\_attr\_create  
     lbm.h, 423  
 lbm\_context\_attr\_create\_default  
     lbm.h, 424  
 lbm\_context\_attr\_create\_from\_xml  
     lbm.h, 424  
 lbm\_context\_attr\_delete  
     lbm.h, 424  
 lbm\_context\_attr\_dump  
     lbm.h, 425  
 lbm\_context\_attr\_dup  
     lbm.h, 425  
 lbm\_context\_attr\_getopt  
     lbm.h, 425  
 lbm\_context\_attr\_option\_size  
     lbm.h, 426  
 lbm\_context\_attr\_set\_from\_xml  
     lbm.h, 426  
 lbm\_context\_attr\_setopt  
     lbm.h, 426  
 lbm\_context\_attr\_str\_getopt  
     lbm.h, 427  
 lbm\_context\_attr\_str\_setopt  
     lbm.h, 427  
 lbm\_context\_create  
     lbm.h, 428  
 lbm\_context\_delete  
     lbm.h, 428  
 lbm\_context\_delete\_ex  
     lbm.h, 429  
 lbm\_context\_dump  
     lbm.h, 429  
 lbm\_context\_event\_cb\_proc  
     lbm.h, 385  
 lbm\_context\_event\_func\_t  
     lbm.h, 386  
 lbm\_context\_event\_func\_t\_stct, 118  
 LBM\_CONTEXT\_EVENT\_UMQ\_-  
     INSTANCE\_LIST\_-  
     NOTIFICATION  
         lbm.h, 358  
         LBM\_CONTEXT\_EVENT\_UMQ\_-  
             REGISTRATION\_-  
             COMPLETE\_EX  
                 lbm.h, 358  
         LBM\_CONTEXT\_EVENT\_UMQ\_-  
             REGISTRATION\_-  
             COMPLETE\_EX\_FLAG\_-  
             QUORUM  
                 lbm.h, 358  
         lbm\_context\_event\_umq\_registration\_-  
             complete\_ex\_t  
                 lbm.h, 386  
         lbm\_context\_event\_umq\_registration\_-  
             complete\_ex\_t\_stct, 119  
                 flags, 119  
                 queue, 119  
                 queue\_id, 119  
                 registration\_id, 119  
         LBM\_CONTEXT\_EVENT\_UMQ\_-  
             REGISTRATION\_ERROR  
                 lbm.h, 358  
         lbm\_context\_event\_umq\_registration\_-  
             ex\_t  
                 lbm.h, 386  
         lbm\_context\_event\_umq\_registration\_-  
             ex\_t\_stct, 121  
                 flags, 121  
                 queue, 122  
                 queue\_id, 121  
                 queue\_instance, 121  
                 queue\_instance\_index, 121  
                 registration\_id, 121  
         LBM\_CONTEXT\_EVENT\_UMQ\_-  
             REGISTRATION\_-  
             SUCCESS\_EX  
                 lbm.h, 358  
         lbm\_context\_from\_rcv  
             lbm.h, 429  
         lbm\_context\_from\_src  
             lbm.h, 429  
         lbm\_context\_from\_wildcard\_rcv  
             lbm.h, 430  
         lbm\_context\_get\_name  
             lbm.h, 430  
         lbm\_context\_getopt

lbm.h, 430  
lmb\_context\_lbtipc\_unblock  
    lmb.h, 431  
lmb\_context\_process\_events  
    lmb.h, 431  
lmb\_context\_process\_lbtipc\_messages  
    lmb.h, 431  
lmb\_context\_recv\_immediate\_msgs  
    lmb.h, 432  
lmb\_context\_recv\_immediate\_msgs\_-  
    func\_t  
    lmb.h, 386  
lmb\_context\_recv\_immediate\_msgs\_-  
    func\_t\_stct, 123  
lmb\_context\_recv\_immediate\_topic\_msgs  
    lmb.h, 432  
lmb\_context\_reactor\_only\_create  
    lmb.h, 433  
lmb\_context\_reset\_im\_rcv\_transport\_-  
    stats  
    lmb.h, 433  
lmb\_context\_reset\_im\_src\_transport\_-  
    stats  
    lmb.h, 434  
lmb\_context\_reset\_rcv\_transport\_stats  
    lmb.h, 434  
lmb\_context\_reset\_src\_transport\_stats  
    lmb.h, 434  
lmb\_context\_reset\_stats  
    lmb.h, 434  
lmb\_context\_retrieve\_im\_rcv\_transport\_-  
    stats  
    lmb.h, 434  
lmb\_context\_retrieve\_im\_src\_transport\_-  
    stats  
    lmb.h, 435  
lmb\_context\_retrieve\_rcv\_transport\_stats  
    lmb.h, 435  
lmb\_context\_retrieve\_src\_transport\_stats  
    lmb.h, 436  
lmb\_context\_retrieve\_stats  
    lmb.h, 436  
lmb\_context\_set\_name  
    lmb.h, 437  
lmb\_context\_setopt  
    lmb.h, 437  
lmb\_context\_src\_cb\_proc  
    lmb.h, 386  
lmb\_context\_src\_event\_func\_t  
    lmb.h, 387  
lmb\_context\_src\_event\_func\_t\_stct, 124  
lmb\_context\_stats\_t  
    lmb.h, 387  
lmb\_context\_stats\_t\_stct, 125  
    lbtrm\_unknown\_msgs\_rcved, 127  
    lbtru\_unknown\_msgs\_rcved, 127  
    resp\_blocked, 127  
    resp\_would\_block, 128  
    send\_blocked, 127  
    send\_would\_block, 127  
    tr\_bytes\_rcved, 126  
    tr\_bytes\_sent, 125  
    tr\_dgrams\_dropped\_malformed, 126  
    tr\_dgrams\_dropped\_type, 126  
    tr\_dgrams\_dropped\_ver, 126  
    tr\_dgrams\_rcved, 126  
    tr\_dgrams\_send\_failed, 126  
    tr\_dgrams\_sent, 125  
    tr\_rcv\_topics, 127  
    tr\_rcv\_unresolved\_topics, 127  
    tr\_src\_topics, 126  
    uim\_dup\_msgs\_rcved, 128  
    uim\_msgs\_no\_stream\_rcved, 128  
lmb\_context\_str\_getopt  
    lmb.h, 437  
lmb\_context\_str\_setopt  
    lmb.h, 438  
lmb\_context\_topic\_resolution\_request  
    lmb.h, 438  
lmb\_context Unblock  
    lmb.h, 439  
lmb\_create\_random\_id  
    lmb.h, 439  
lmb\_ctx\_umq\_get\_inflight  
    lmb.h, 439  
lmb\_ctx\_umq\_queue\_topic\_list  
    lmb.h, 440  
lmb\_ctx\_umq\_queue\_topic\_list\_info\_t,  
    129  
        num\_topics, 129  
        topics, 129  
lmb\_daemon\_event\_cb\_proc

lbm.h, 387  
**LBM\_DAEMON\_EVENT\_-CONNECT\_ERROR**  
 lbm.h, 359  
**LBM\_DAEMON\_EVENT\_-CONNECT\_TIMEOUT**  
 lbm.h, 359  
**LBM\_DAEMON\_EVENT\_-CONNECTED**  
 lbm.h, 359  
**LBM\_DAEMON\_EVENT\_-DISCONNECTED**  
 lbm.h, 359  
**lbm\_debug\_dump**  
 lbm.h, 440  
**lbm\_debug\_filename**  
 lbm.h, 440  
**lbm\_debug\_mask**  
 lbm.h, 441  
**lbm\_delete\_cb\_info\_t\_stct**, 130  
 cbproc, 130  
 clientd, 130  
**lbm\_delete\_cb\_proc**  
 lbmht.h, 539  
**lbm\_deserialize\_response**  
 lbm.h, 441  
**LBM\_EDAEMONCONN**  
 lbm.h, 359  
**LBM\_EINPROGRESS**  
 lbm.h, 359  
**LBM\_EINVAL**  
 lbm.h, 359  
**LBM\_EMSG\_SELECTOR**  
 lbm.h, 359  
**LBM\_ENO\_QUEUE\_REG**  
 lbm.h, 359  
**LBM\_ENO\_STORE\_REG**  
 lbm.h, 360  
**LBM\_ENOMEM**  
 lbm.h, 360  
**LBM\_EOP**  
 lbm.h, 360  
**LBM\_EOPNOTSUPP**  
 lbm.h, 360  
**LBM\_EOS**  
 lbm.h, 360

**lbm\_errmsg**  
 lbm.h, 441  
**lbm\_errnum**  
 lbm.h, 441  
**LBM\_ETIMEDOUT**  
 lbm.h, 360  
**LBM\_EUMENOREG**  
 lbm.h, 360  
**lbm\_event\_dispatch**  
 lbm.h, 442  
**lbm\_event\_dispatch\_unblock**  
 lbm.h, 442  
**lbm\_event\_queue\_attr\_create**  
 lbm.h, 442  
**lbm\_event\_queue\_attr\_create\_default**  
 lbm.h, 443  
**lbm\_event\_queue\_attr\_create\_from\_xml**  
 lbm.h, 443  
**lbm\_event\_queue\_attr\_delete**  
 lbm.h, 444  
**lbm\_event\_queue\_attr\_dump**  
 lbm.h, 444  
**lbm\_event\_queue\_attr\_dup**  
 lbm.h, 444  
**lbm\_event\_queue\_attr\_getopt**  
 lbm.h, 445  
**lbm\_event\_queue\_attr\_option\_size**  
 lbm.h, 445  
**lbm\_event\_queue\_attr\_set\_from\_xml**  
 lbm.h, 445  
**lbm\_event\_queue\_attr\_setopt**  
 lbm.h, 446  
**lbm\_event\_queue\_attr\_str\_getopt**  
 lbm.h, 446  
**lbm\_event\_queue\_attr\_str\_setopt**  
 lbm.h, 446  
**LBM\_EVENT\_QUEUE\_BLOCK**  
 lbm.h, 360  
**lbm\_event\_queue\_cancel\_cb\_info\_t\_stct**,  
 131  
 cbproc, 131  
 clientd, 131  
 event\_queue, 131  
**lbm\_event\_queue\_cancel\_cb\_proc**  
 lbm.h, 387  
**lbm\_event\_queue\_create**

lbtm.h, 447  
LBM\_EVENT\_QUEUE\_DELAY\_-  
    WARNING  
        lbtm.h, 360  
lbtm\_event\_queue\_delete  
    lbtm.h, 447  
lbtm\_event\_queue\_dump  
    lbtm.h, 448  
LBM\_EVENT\_QUEUE\_ENQUEUE\_-  
    NOTIFICATION  
        lbtm.h, 360  
lbtm\_event\_queue\_from\_rcv  
    lbtm.h, 448  
lbtm\_event\_queue\_from\_src  
    lbtm.h, 448  
lbtm\_event\_queue\_from\_wildcard\_rcv  
    lbtm.h, 448  
lbtm\_event\_queue\_getopt  
    lbtm.h, 449  
lbtm\_event\_queue\_monitor\_proc  
    lbtm.h, 388  
LBM\_EVENT\_QUEUE\_POLL  
    lbtm.h, 361  
lbtm\_event\_queue\_reset\_stats  
    lbtm.h, 449  
lbtm\_event\_queue\_retrieve\_stats  
    lbtm.h, 449  
lbtm\_event\_queue\_setopt  
    lbtm.h, 450  
lbtm\_event\_queue\_shutdown  
    lbtm.h, 450  
lbtm\_event\_queue\_size  
    lbtm.h, 450  
LBM\_EVENT\_QUEUE\_SIZE\_-  
    WARNING  
        lbtm.h, 361  
lbtm\_event\_queue\_stats\_t  
    lbtm.h, 389  
lbtm\_event\_queue\_stats\_t\_stct, 132  
    age\_max, 142  
    age\_mean, 141  
    age\_min, 141  
    callback\_events, 142  
    callback\_events\_svc\_max, 142  
    callback\_events\_svc\_mean, 142  
    callback\_events\_svc\_min, 142  
    callback\_events\_tot, 142  
cancel\_events, 139  
cancel\_events\_svc\_max, 140  
cancel\_events\_svc\_mean, 140  
cancel\_events\_svc\_min, 140  
cancel\_events\_tot, 139  
context\_source\_events, 140  
context\_source\_events\_svc\_max,  
    141  
context\_source\_events\_svc\_mean,  
    141  
context\_source\_events\_svc\_min,  
    140  
context\_source\_events\_tot, 140  
data\_msgs, 133  
data\_msgs\_svc\_max, 134  
data\_msgs\_svc\_mean, 134  
data\_msgs\_svc\_min, 133  
data\_msgs\_tot, 133  
events, 141  
events\_tot, 141  
io\_events, 137  
io\_events\_svc\_max, 137  
io\_events\_svc\_mean, 137  
io\_events\_svc\_min, 137  
io\_events\_tot, 137  
resp\_msgs, 134  
resp\_msgs\_svc\_max, 135  
resp\_msgs\_svc\_mean, 134  
resp\_msgs\_svc\_min, 134  
resp\_msgs\_tot, 134  
source\_events, 138  
source\_events\_svc\_max, 139  
source\_events\_svc\_mean, 139  
source\_events\_svc\_min, 139  
source\_events\_tot, 138  
timer\_events, 137  
timer\_events\_svc\_max, 138  
timer\_events\_svc\_mean, 138  
timer\_events\_svc\_min, 138  
timer\_events\_tot, 138  
topicless\_im\_msgs, 135  
topicless\_im\_msgs\_svc\_max, 136  
topicless\_im\_msgs\_svc\_mean, 135  
topicless\_im\_msgs\_svc\_min, 135  
topicless\_im\_msgs\_tot, 135

unblock\_events, 139  
 unblock\_events\_tot, 139  
 wrcv\_msgs, 136  
 wrcv\_msgs\_svc\_max, 136  
 wrcv\_msgs\_svc\_mean, 136  
 wrcv\_msgs\_svc\_min, 136  
 wrcv\_msgs\_tot, 136  
 lbm\_event\_queue\_str\_getopt  
     lbtm.h, 451  
 lbm\_event\_queue\_str\_setopt  
     lbtm.h, 451  
**LBM\_EWOULDBLOCK**  
     lbtm.h, 361  
 lbm\_fd\_cb\_proc  
     lbtm.h, 389  
**LBM\_FD\_EVENT\_ACCEPT**  
     lbtm.h, 361  
**LBM\_FD\_EVENT\_ALL**  
     lbtm.h, 361  
**LBM\_FD\_EVENT\_CLOSE**  
     lbtm.h, 361  
**LBM\_FD\_EVENT\_CONNECT**  
     lbtm.h, 361  
**LBM\_FD\_EVENT\_EXCEPT**  
     lbtm.h, 361  
**LBM\_FD\_EVENT\_READ**  
     lbtm.h, 361  
**LBM\_FD\_EVENT\_WRITE**  
     lbtm.h, 361  
 lbm\_flight\_size\_inflight\_t\_stct, 144  
     bytes, 144  
     messages, 144  
 lbm\_flight\_size\_set\_inflight\_cb\_proc  
     lbtm.h, 390  
 lbm\_flight\_size\_set\_inflight\_ex\_cb\_proc  
     lbtm.h, 390  
**LBM\_FLIGHT\_SIZE\_TYPE\_UBL**  
     lbtm.h, 362  
**LBM\_FLIGHT\_SIZE\_TYPE\_UME**  
     lbtm.h, 362  
**LBM\_FLIGHT\_SIZE\_TYPE\_UMQ**  
     lbtm.h, 362  
 lbm\_get\_jms\_msg\_id  
     lbtm.h, 451  
 lbm\_hf\_rcv\_create  
     lbtm.h, 451  
 lbm\_hf\_rcv\_delete  
     lbtm.h, 452  
 lbm\_hf\_rcv\_delete\_ex  
     lbtm.h, 453  
 lbm\_hf\_rcv\_from\_recv  
     lbtm.h, 453  
 lbm\_hf\_rcv\_topic\_dump  
     lbtm.h, 453  
**lbtm\_hf\_sequence\_number\_t\_stct**, 145  
     u32, 145  
     u64, 145  
 lbm\_hf\_src\_create  
     lbtm.h, 454  
 lbm\_hf\_src\_send  
     lbtm.h, 454  
 lbm\_hf\_src\_send\_ex  
     lbtm.h, 455  
 lbm\_hf\_src\_send\_recv\_reset  
     lbtm.h, 456  
 lbm\_hf\_src\_sendv  
     lbtm.h, 457  
 lbm\_hf\_src\_sendv\_ex  
     lbtm.h, 458  
 lbm\_hfx\_attr\_create  
     lbtm.h, 459  
 lbm\_hfx\_attr\_create\_default  
     lbtm.h, 459  
 lbm\_hfx\_attr\_create\_from\_xml  
     lbtm.h, 459  
 lbm\_hfx\_attr\_delete  
     lbtm.h, 460  
 lbm\_hfx\_attr\_dump  
     lbtm.h, 460  
 lbm\_hfx\_attr\_dup  
     lbtm.h, 460  
 lbm\_hfx\_attr\_getopt  
     lbtm.h, 461  
 lbm\_hfx\_attr\_option\_size  
     lbtm.h, 461  
 lbm\_hfx\_attr\_set\_from\_xml  
     lbtm.h, 461  
 lbm\_hfx\_attr\_setopt  
     lbtm.h, 462  
 lbm\_hfx\_attr\_str\_getopt  
     lbtm.h, 462  
 lbm\_hfx\_attr\_str\_setopt

lbm.h, 462  
lbm\_hfx\_create  
    lbm.h, 463  
lbm\_hfx\_delete  
    lbm.h, 463  
lbm\_hfx\_delete\_ex  
    lbm.h, 464  
lbm\_hfx\_dump  
    lbm.h, 464  
lbm\_hfx\_getopt  
    lbm.h, 465  
lbm\_hfx\_rcv\_create  
    lbm.h, 465  
lbm\_hfx\_rcv\_delete  
    lbm.h, 465  
lbm\_hfx\_rcv\_delete\_ex  
    lbm.h, 466  
lbm\_hfx\_rcv\_topic\_dump  
    lbm.h, 466  
lbm\_hfx\_setopt  
    lbm.h, 467  
lbm\_hfx\_str\_getopt  
    lbm.h, 467  
lbm\_hfx\_str\_setopt  
    lbm.h, 467  
lbm\_hypertopic\_rcv\_add  
    lbumht.h, 540  
lbm\_hypertopic\_rcv\_cb\_proc  
    lbumht.h, 539  
lbm\_hypertopic\_rcv\_delete  
    lbumht.h, 540  
lbm\_hypertopic\_rcv\_destroy  
    lbumht.h, 540  
lbm\_hypertopic\_rcv\_init  
    lbumht.h, 541  
lbm\_immediate\_msg\_cb\_proc  
    lbm.h, 390  
lbm\_iovec\_t  
    lbm.h, 391  
lbm\_iovec\_t\_stct, 146  
    iov\_base, 146  
    iov\_len, 146  
lbm\_ip4\_address\_mask\_t  
    lbm.h, 391  
lbm\_ip4\_address\_mask\_t\_stct, 147  
    addr, 147  
            bits, 147  
lbm\_is\_ume\_capable  
    lbm.h, 468  
lbm\_is\_umq\_capable  
    lbm.h, 468  
lbm\_license\_file  
    lbm.h, 468  
lbm\_license\_str  
    lbm.h, 468  
lbm\_log  
    lbm.h, 469  
LBM\_LOG\_ALERT  
    lbm.h, 362  
lbm\_log\_cb\_proc  
    lbm.h, 391  
LBM\_LOG\_CRIT  
    lbm.h, 362  
LBM\_LOG\_DEBUG  
    lbm.h, 362  
LBM\_LOG\_EMERG  
    lbm.h, 362  
LBM\_LOG\_ERR  
    lbm.h, 362  
LBM\_LOG\_INFO  
    lbm.h, 362  
LBM\_LOG\_NOTICE  
    lbm.h, 362  
LBM\_LOG\_WARNING  
    lbm.h, 363  
lbm\_logf  
    lbm.h, 469  
lbm\_mim\_unrecloss\_func\_t  
    lbm.h, 392  
lbm\_mim\_unrecloss\_func\_t\_stct, 148  
lbm\_mim\_unrecloss\_function\_cb  
    lbm.h, 392  
LBM\_MSG\_BOS  
    lbm.h, 363  
lbm\_msg\_channel\_info\_t  
    lbm.h, 392  
lbm\_msg\_channel\_info\_t\_stct, 149  
    channel\_number, 149  
    flags, 149  
LBM\_MSG\_COMPLETE\_BATCH  
    lbm.h, 363  
LBM\_MSG\_DATA

lbm.h, 363  
 lbm\_msg\_delete  
     lbm.h, 469  
 LBM\_MSG\_END\_BATCH  
     lbm.h, 363  
 LBM\_MSG\_EOS  
     lbm.h, 363  
 lbm\_msg\_extract\_ume\_ack  
     lbm.h, 470  
 LBM\_MSG\_FLAG\_DELIVERY\_-  
     LATENCY  
     lbm.h, 363  
 LBM\_MSG\_FLAG\_END\_BATCH  
     lbm.h, 363  
 LBM\_MSG\_FLAG\_HF\_32  
     lbm.h, 363  
 LBM\_MSG\_FLAG\_HF\_64  
     lbm.h, 364  
 LBM\_MSG\_FLAG\_HF\_DUPLICATE  
     lbm.h, 364  
 LBM\_MSG\_FLAG\_HF\_OPTIONAL  
     lbm.h, 364  
 LBM\_MSG\_FLAG\_HF\_PASS\_-  
     THROUGH  
     lbm.h, 364  
 LBM\_MSG\_FLAG\_IMMEDIATE  
     lbm.h, 364  
 LBM\_MSG\_FLAG\_NUMBERED\_-  
     CHANNEL  
     lbm.h, 364  
 LBM\_MSG\_FLAG\_OTR  
     lbm.h, 364  
 LBM\_MSG\_FLAG\_RETRANSMIT  
     lbm.h, 364  
 LBM\_MSG\_FLAG\_START\_BATCH  
     lbm.h, 364  
 LBM\_MSG\_FLAG\_TOPICLESS  
     lbm.h, 364  
 LBM\_MSG\_FLAG\_UME\_-  
     RETRANSMIT  
     lbm.h, 365  
 LBM\_MSG\_FLAG\_UMQ\_-  
     REASSIGNED  
     lbm.h, 365  
 LBM\_MSG\_FLAG\_UMQ\_-  
     RESUBMITTED

lbm.h, 365  
 LBM\_MSG\_FLUSH  
     lbm.h, 365  
 lbm\_msg\_fragment\_info\_t  
     lbm.h, 392  
 lbm\_msg\_fragment\_info\_t\_stct, 150  
     offset, 150  
     start\_sequence\_number, 150  
     total\_message\_length, 150  
 lbm\_msg\_gateway\_info\_t  
     lbm.h, 392  
 lbm\_msg\_gateway\_info\_t\_stct, 151  
     sequence\_number, 151  
     source, 151  
 LBM\_MSG\_HF\_RESET  
     lbm.h, 365  
 LBM\_MSG iov\_GATHER  
     lbm.h, 365  
 lbm\_msg\_is\_fragment  
     lbm.h, 470  
 LBM\_MSG\_NO\_SOURCE\_-  
     NOTIFICATION  
     lbm.h, 365  
 lbm\_msg\_properties\_clear  
     lbm.h, 470  
 lbm\_msg\_properties\_create  
     lbm.h, 470  
 lbm\_msg\_properties\_delete  
     lbm.h, 471  
 lbm\_msg\_properties\_get  
     lbm.h, 471  
 lbm\_msg\_properties\_iter\_create  
     lbm.h, 472  
 lbm\_msg\_properties\_iter\_delete  
     lbm.h, 472  
 lbm\_msg\_properties\_iter\_first  
     lbm.h, 473  
 lbm\_msg\_properties\_iter\_next  
     lbm.h, 473  
 lbm\_msg\_properties\_iter\_t\_stct, 152  
 LBM\_MSG\_PROPERTIES\_MAX\_-  
     NAMELEN  
     lbm.h, 365  
 lbm\_msg\_properties\_set  
     lbm.h, 474  
 LBM\_MSG\_PROPERTY\_BOOLEAN

lbm.h, 366  
LBM\_MSG\_PROPERTY\_BYTE  
  lbm.h, 366  
LBM\_MSG\_PROPERTY\_DOUBLE  
  lbm.h, 366  
LBM\_MSG\_PROPERTY\_FLOAT  
  lbm.h, 366  
LBM\_MSG\_PROPERTY\_INT  
  lbm.h, 366  
LBM\_MSG\_PROPERTY\_LONG  
  lbm.h, 366  
LBM\_MSG\_PROPERTY\_NONE  
  lbm.h, 366  
LBM\_MSG\_PROPERTY\_SHORT  
  lbm.h, 366  
LBM\_MSG\_PROPERTY\_STRING  
  lbm.h, 366  
LBM\_MSG\_REQUEST  
  lbm.h, 366  
LBM\_MSG\_RESPONSE  
  lbm.h, 367  
lbm\_msg\_retain  
  lbm.h, 474  
lbm\_msg\_retrieve\_fragment\_info  
  lbm.h, 475  
lbm\_msg\_retrieve\_gateway\_info  
  lbm.h, 475  
lbm\_msg\_retrievemsgid  
  lbm.h, 475  
lbm\_msg\_retrieve\_umq\_index  
  lbm.h, 475  
LBM\_MSG\_START\_BATCH  
  lbm.h, 367  
lbm\_msg\_t\_stct, 153  
  channel\_info, 156  
  copied\_state, 154  
  data, 156  
  flags, 155  
  hf\_sequence\_number, 157  
  len, 156  
  properties, 157  
  response, 156  
  sequence\_number, 156  
  source, 154  
  source\_clientd, 156  
  topic\_name, 154  
    type, 154  
lbm\_msg\_ume\_can\_send\_explicit\_ack  
  lbm.h, 476  
LBM\_MSG\_UME\_-  
  Deregistration\_-  
  COMPLETE\_EX  
  lbm.h, 367  
lbm\_msg\_ume\_deregistration\_ex\_t  
  lbm.h, 393  
lbm\_msg\_ume\_deregistration\_ex\_t\_stct,  
  158  
  flags, 158  
  rcv\_registration\_id, 158  
  sequence\_number, 158  
  src\_registration\_id, 158  
  store, 159  
  store\_index, 159  
LBM\_MSG\_UME\_-  
  Deregistration\_-  
  SUCCESS\_EX  
  lbm.h, 367  
LBM\_MSG\_UME\_-  
  Deregistration\_-  
  SUCCESS\_EX\_FLAG\_RPP  
  lbm.h, 367  
LBM\_MSG\_UME\_REGISTRATION\_-  
  CHANGE  
  lbm.h, 367  
LBM\_MSG\_UME\_REGISTRATION\_-  
  COMPLETE\_EX  
  lbm.h, 367  
LBM\_MSG\_UME\_REGISTRATION\_-  
  COMPLETE\_EX\_FLAG\_-  
  QUORUM  
  lbm.h, 367  
LBM\_MSG\_UME\_REGISTRATION\_-  
  COMPLETE\_EX\_FLAG\_-  
  RXREQMAX  
  lbm.h, 368  
lbm\_msg\_ume\_registration\_complete\_-  
  ex\_t  
  lbm.h, 393  
lbm\_msg\_ume\_registration\_complete\_-  
  ex\_t\_stct, 160  
  flags, 160  
  sequence\_number, 160

LBM\_MSG\_UME\_REGISTRATION\_-  
 ERROR  
 lbm.h, 368

lbm\_msg\_ume\_registration\_ex\_t  
 lbm.h, 393

lbm\_msg\_ume\_registration\_ex\_t\_stct,  
 161  
 flags, 161  
 rcv\_registration\_id, 161  
 sequence\_number, 161  
 src\_registration\_id, 161  
 store, 162  
 store\_index, 161

LBM\_MSG\_UME\_REGISTRATION\_-  
 SUCCESS  
 lbm.h, 368

LBM\_MSG\_UME\_REGISTRATION\_-  
 SUCCESS\_EX  
 lbm.h, 368

LBM\_MSG\_UME\_REGISTRATION\_-  
 SUCCESS\_EX\_FLAG\_-  
 NOCACHE  
 lbm.h, 368

LBM\_MSG\_UME\_REGISTRATION\_-  
 SUCCESS\_EX\_FLAG\_OLD  
 lbm.h, 368

LBM\_MSG\_UME\_REGISTRATION\_-  
 SUCCESS\_EX\_FLAG\_RPP  
 lbm.h, 368

lbm\_msg\_ume\_registration\_t  
 lbm.h, 393

lbm\_msg\_ume\_registration\_t\_stct, 163  
 rcv\_registration\_id, 163  
 src\_registration\_id, 163

lbm\_msg\_ume\_send\_explicit\_ack  
 lbm.h, 476

LBM\_MSG\_UMQ\_-  
 Deregistration\_-  
 Complete\_Ex  
 lbm.h, 368

LBM\_MSG\_UMQ\_-  
 Deregistration\_-  
 Complete\_Ex\_Flag\_Ulb  
 lbm.h, 369

lbm\_msg\_umq\_deregistration\_-  
 complete\_ex\_t

lbm.h, 393  
 complete\_ex\_t\_stct, 164  
 flags, 164  
 queue\_id, 164

LBM\_MSG\_UMQ\_INDEX\_-  
 Assigned\_Ex  
 lbm.h, 369

LBM\_MSG\_UMQ\_INDEX\_-  
 Assigned\_Ex\_Flag\_-  
 Requested  
 lbm.h, 369

LBM\_MSG\_UMQ\_INDEX\_-  
 Assigned\_Ex\_Flag\_Ulb  
 lbm.h, 369

lbm\_msg\_umq\_index\_assigned\_ex\_t  
 lbm.h, 393

lbm\_msg\_umq\_index\_assigned\_ex\_t\_-  
 stct, 165  
 flags, 165  
 queue\_id, 165

LBM\_MSG\_UMQ\_INDEX\_-  
 Assignment\_-  
 Eligibility\_Error  
 lbm.h, 369

LBM\_MSG\_UMQ\_INDEX\_-  
 Assignment\_-  
 Eligibility\_Start\_-  
 Complete\_Ex  
 lbm.h, 369

LBM\_MSG\_UMQ\_INDEX\_-  
 Assignment\_-  
 Eligibility\_Start\_-  
 Complete\_Ex\_Flag\_Ulb  
 lbm.h, 369

lbm\_msg\_umq\_index\_assignment\_-  
 eligibility\_start\_complete\_ex\_t  
 lbm.h, 393

lbm\_msg\_umq\_index\_assignment\_-  
 eligibility\_start\_complete\_ex\_-  
 t\_stct, 166  
 flags, 166  
 queue\_id, 166

LBM\_MSG\_UMQ\_INDEX\_-  
 Assignment\_-  
 Eligibility\_Stop\_-

COMPLETE\_EX  
lbtipc.h, 369

LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNMENT\_-  
ELIGIBILITY\_STOP\_-  
COMPLETE\_EX\_FLAG\_ULB  
lbtipc.h, 370

lbtipc\_msg\_umq\_index\_assignment\_-  
eligibility\_stop\_complete\_ex\_t  
lbtipc.h, 394

lbtipc\_msg\_umq\_index\_assignment\_-  
eligibility\_stop\_complete\_ex\_t\_stct, 167  
flags, 167  
queue\_id, 167

LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNMENT\_ERROR  
lbtipc.h, 370

LBM\_MSG\_UMQ\_INDEX\_-  
RELEASED\_EX  
lbtipc.h, 370

LBM\_MSG\_UMQ\_INDEX\_-  
RELEASED\_EX\_FLAG\_ULB  
lbtipc.h, 370

lbtipc\_msg\_umq\_index\_released\_ex\_t  
lbtipc.h, 394

lbtipc\_msg\_umq\_index\_released\_ex\_t\_stct, 168  
flags, 168  
queue\_id, 168

lbtipc\_msg\_umq\_reassign  
lbtipc.h, 476

LBM\_MSG\_UMQ\_REASSIGN\_-  
FLAG\_DISCARD  
lbtipc.h, 370

LBM\_MSG\_UMQ\_REGISTRATION\_-  
COMPLETE\_EX  
lbtipc.h, 370

LBM\_MSG\_UMQ\_REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_-  
QUORUM  
lbtipc.h, 370

LBM\_MSG\_UMQ\_REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_ULB  
lbtipc.h, 370

lbtipc\_msg\_umq\_registration\_complete\_-  
ex\_t  
lbtipc.h, 394

lbtipc\_msg\_umq\_registration\_complete\_-  
ex\_t\_stct, 169  
assignment\_id, 169  
flags, 169  
queue, 169  
queue\_id, 169

LBM\_MSG\_UMQ\_REGISTRATION\_-  
ERROR  
lbtipc.h, 371

LBM\_MSG\_UNRECOVERABLE\_-  
LOSS  
lbtipc.h, 371

LBM\_MSG\_UNRECOVERABLE\_-  
LOSS\_BURST  
lbtipc.h, 371

lbtipc\_msgs\_no\_topic\_rcved  
lbtipc\_rcv\_transport\_stats\_lbtipc\_t\_-  
stct, 173

lbtipc\_rcv\_transport\_stats\_lbtrdma\_-  
t\_stct, 175

lbtipc\_rcv\_transport\_stats\_lbtrm\_t\_-  
stct, 181

lbtipc\_rcv\_transport\_stats\_lbtru\_t\_-  
stct, 188

lbtipc\_rcv\_transport\_stats\_tcp\_t\_stct,  
192

lbtipc\_msgs\_rcved  
lbtipc\_rcv\_transport\_stats\_lbtipc\_t\_-  
stct, 173

lbtipc\_rcv\_transport\_stats\_lbtrdma\_-  
t\_stct, 175

lbtipc\_rcv\_transport\_stats\_lbtrm\_t\_-  
stct, 181

lbtipc\_rcv\_transport\_stats\_lbtru\_t\_-  
stct, 188

lbtipc\_rcv\_transport\_stats\_tcp\_t\_stct,  
192

lbtipc\_multicast\_immediate\_message  
lbtipc.h, 477

lbtipc\_multicast\_immediate\_request  
lbtipc.h, 477

lbtipc\_queue\_immediate\_message  
lbtipc.h, 478

LBM\_RCV\_BLOCK  
 lbm.h, 371

LBM\_RCV\_BLOCK\_TEMP  
 lbm.h, 371

lbm\_rcv\_cb\_proc  
 lbm.h, 394

lbm\_rcv\_create  
 lbm.h, 478

lbm\_rcv\_delete  
 lbm.h, 479

lbm\_rcv\_delete\_ex  
 lbm.h, 480

lbm\_rcv\_from\_hf\_rcv  
 lbm.h, 480

lbm\_rcv\_from\_hfx\_rcv  
 lbm.h, 480

lbm\_rcv\_getopt  
 lbm.h, 480

lbm\_rcv\_msg\_source\_clientd  
 lbm.h, 481

LBM\_RCV\_NONBLOCK  
 lbm.h, 371

lbm\_rcv\_reset\_all\_transport\_stats  
 lbm.h, 481

lbm\_rcv\_reset\_transport\_stats  
 lbm.h, 481

lbm\_rcv\_retrieve\_all\_transport\_stats  
 lbm.h, 482

lbm\_rcv\_retrieve\_transport\_stats  
 lbm.h, 482

lbm\_rcv\_setopt  
 lbm.h, 482

lbm\_rcv\_src\_notification\_create\_function\_cb  
 lbm.h, 395

lbm\_rcv\_src\_notification\_delete\_function\_cb  
 lbm.h, 395

lbm\_rcv\_src\_notification\_func\_t  
 lbm.h, 396

lbm\_rcv\_src\_notification\_func\_t\_stct,  
 171

lbm\_rcv\_str\_getopt  
 lbm.h, 483

lbm\_rcv\_str\_setopt  
 lbm.h, 483

lbm\_rcv\_subscribe\_channel  
 lbm.h, 484

lbm\_rcv\_topic\_attr\_create  
 lbm.h, 484

lbm\_rcv\_topic\_attr\_create\_default  
 lbm.h, 485

lbm\_rcv\_topic\_attr\_create\_from\_xml  
 lbm.h, 485

lbm\_rcv\_topic\_attr\_delete  
 lbm.h, 485

lbm\_rcv\_topic\_attr\_dump  
 lbm.h, 486

lbm\_rcv\_topic\_attr\_dup  
 lbm.h, 486

lbm\_rcv\_topic\_attr\_getopt  
 lbm.h, 486

lbm\_rcv\_topic\_attr\_option\_size  
 lbm.h, 487

lbm\_rcv\_topic\_attr\_set\_from\_xml  
 lbm.h, 487

lbm\_rcv\_topic\_attr\_setopt  
 lbm.h, 487

lbm\_rcv\_topic\_attr\_str\_getopt  
 lbm.h, 488

lbm\_rcv\_topic\_attr\_str\_setopt  
 lbm.h, 488

lbm\_rcv\_topic\_dump  
 lbm.h, 489

lbm\_rcv\_topic\_lookup  
 lbm.h, 489

lbm\_rcv\_transport\_stats\_daemon\_t  
 lbm.h, 396

lbm\_rcv\_transport\_stats\_daemon\_t\_stct,  
 172

bytes\_rcved, 172

lbm\_rcv\_transport\_stats\_lbtipc\_t\_stct,  
 173

bytes\_rcved, 173

lbm\_msgs\_no\_topic\_rcved, 173

lbm\_msgs\_rcved, 173

lbm\_reqs\_rcved, 174

msgs\_rcved, 173

lbm\_rcv\_transport\_stats\_lbtrdma\_t\_stct,  
 175

bytes\_rcved, 175

lbm\_msgs\_no\_topic\_rcved, 175



msgs, 194  
 num\_msgs, 194  
**lbm\_rcv\_umq\_queue\_msg\_retrieve**  
 lbm.h, 492  
**lbm\_rcv\_umq\_queue\_msg\_retrieve\_-  
 info\_t**, 195  
 msgs, 195  
 num\_msgs, 195  
**lbm\_rcv\_unsubscribe\_channel**  
 lbm.h, 493  
**lbm\_rcv\_unsubscribe\_channel\_ex**  
 lbm.h, 493  
**lbm\_register\_fd**  
 lbm.h, 494  
**lbm\_reqs\_rcved**  
 lbm\_rcv\_transport\_stats\_lbtpc\_t\_-  
 stct, 174  
 lbm\_rcv\_transport\_stats\_lbtrdma\_-  
 t\_stct, 176  
 lbm\_rcv\_transport\_stats\_lbtrm\_t\_-  
 stct, 181  
 lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
 stct, 188  
 lbm\_rcv\_transport\_stats\_tcp\_t\_stct,  
 192  
**lbm\_request\_cb\_proc**  
 lbm.h, 396  
**lbm\_request\_delete**  
 lbm.h, 494  
**lbm\_request\_delete\_ex**  
 lbm.h, 495  
**lbm\_response\_delete**  
 lbm.h, 495  
**lbm\_schedule\_timer**  
 lbm.h, 495  
**lbm\_schedule\_timer\_recurring**  
 lbm.h, 496  
**lbm\_send\_request**  
 lbm.h, 497  
**lbm\_send\_request\_ex**  
 lbm.h, 497  
**lbm\_send\_response**  
 lbm.h, 498  
**lbm\_serialize\_response**  
 lbm.h, 499  
**lbm\_serialized\_response\_delete**  
 lbm.h, 499  
**lbm\_serialized\_response\_t\_stct**, 196  
**lbm\_set\_lbtrm\_loss\_rate**  
 lbm.h, 499  
**lbm\_set\_lbtrm\_src\_loss\_rate**  
 lbm.h, 500  
**lbm\_set\_lbtru\_loss\_rate**  
 lbm.h, 500  
**lbm\_set\_lbtru\_src\_loss\_rate**  
 lbm.h, 500  
**lbm\_set\_umm\_info**  
 lbm.h, 500  
**LBM\_SRC\_BLOCK**  
 lbm.h, 371  
**LBM\_SRC\_BLOCK\_TEMP**  
 lbm.h, 371  
**lbm\_src\_cb\_proc**  
 lbm.h, 396  
**lbm\_src\_channel\_create**  
 lbm.h, 501  
**lbm\_src\_channel\_delete**  
 lbm.h, 501  
**lbm\_src\_cost\_func\_t\_stct**, 197  
**lbm\_src\_cost\_function\_cb**  
 lbm.h, 398  
**LBM\_SRC\_COST\_FUNCTION\_-  
 REJECT**  
 lbm.h, 371  
**lbm\_src\_create**  
 lbm.h, 501  
**lbm\_src\_delete**  
 lbm.h, 502  
**lbm\_src\_delete\_ex**  
 lbm.h, 502  
**LBM\_SRC\_EVENT\_CONNECT**  
 lbm.h, 372  
**LBM\_SRC\_EVENT\_DAEMON\_-  
 CONFIRM**  
 lbm.h, 372  
**LBM\_SRC\_EVENT\_DISCONNECT**  
 lbm.h, 372  
**LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_-  
 NOTIFICATION**  
 lbm.h, 372  
**LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_-  
 NOTIFICATION\_STATE\_-**

OVER  
lbtm.h, 372

LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_-  
NOTIFICATION\_STATE\_-  
UNDER  
lbtm.h, 372

lbtm\_src\_event\_flight\_size\_notification\_t  
lbtm.h, 399

lbtm\_src\_event\_flight\_size\_notification\_-  
t\_stct, 198  
state, 198  
type, 198

LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_-  
NOTIFICATION\_TYPE\_ULB  
lbtm.h, 373

LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_-  
NOTIFICATION\_TYPE\_UME  
lbtm.h, 373

LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_-  
NOTIFICATION\_TYPE\_UMQ  
lbtm.h, 373

LBM\_SRC\_EVENT\_SEQUENCE\_-  
NUMBER\_INFO  
lbtm.h, 373

lbtm\_src\_event\_sequence\_number\_info\_t  
lbtm.h, 399

lbtm\_src\_event\_sequence\_number\_info\_-  
t\_stct, 199  
first\_sequence\_number, 199  
flags, 199  
last\_sequence\_number, 199  
msg\_clientd, 199

lbtm\_src\_event\_ume\_ack\_ex\_info\_t  
lbtm.h, 399

lbtm\_src\_event\_ume\_ack\_ex\_info\_t\_stct,  
201  
flags, 201  
msg\_clientd, 202  
rcv\_registration\_id, 201  
sequence\_number, 201  
store, 201  
store\_index, 202

lbtm\_src\_event\_ume\_ack\_info\_t  
lbtm.h, 399

lbtm\_src\_event\_ume\_ack\_info\_t\_stct, 203  
msg\_clientd, 203

rcv\_registration\_id, 203  
sequence\_number, 203

LBM\_SRC\_EVENT\_UME\_-  
DELIVERY\_-  
CONFIRMATION  
lbtm.h, 373

LBM\_SRC\_EVENT\_UME\_-  
DELIVERY\_-  
CONFIRMATION\_EX  
lbtm.h, 373

LBM\_SRC\_EVENT\_UME\_-  
DELIVERY\_-  
CONFIRMATION\_EX\_-  
FLAG\_EXACK  
lbtm.h, 373

LBM\_SRC\_EVENT\_UME\_-  
DELIVERY\_-  
CONFIRMATION\_EX\_-  
FLAG\_OOD  
lbtm.h, 374

LBM\_SRC\_EVENT\_UME\_-  
DELIVERY\_-  
CONFIRMATION\_EX\_-  
FLAG\_UNIQUEACKS  
lbtm.h, 374

LBM\_SRC\_EVENT\_UME\_-  
DELIVERY\_-  
CONFIRMATION\_EX\_-  
FLAG\_UREGID  
lbtm.h, 374

LBM\_SRC\_EVENT\_UME\_-  
DELIVERY\_-  
CONFIRMATION\_EX\_-  
FLAG\_WHOLE\_MESSAGE\_-  
CONFIRMED  
lbtm.h, 374

LBM\_SRC\_EVENT\_UME\_-  
DEREGISTRATION\_-  
COMPLETE\_EX  
lbtm.h, 374

lbtm\_src\_event\_ume\_deregistration\_ex\_t  
lbtm.h, 399

lbtm\_src\_event\_ume\_deregistration\_ex\_-  
t\_stct, 204  
flags, 204  
registration\_id, 204

sequence\_number, 204  
 store, 205  
 store\_index, 204  
**LBM\_SRC\_EVENT\_UME\_-**  
     Deregistration\_Ex  
     Success\_Ex  
     lrbm.h, 374  
**LBM\_SRC\_EVENT\_UME\_-**  
     Message\_Reclaimed  
     lrbm.h, 374  
**LBM\_SRC\_EVENT\_UME\_-**  
     Message\_Reclaimed\_Ex  
     lrbm.h, 375  
**LBM\_SRC\_EVENT\_UME\_-**  
     Message\_Reclaimed\_Ex\_Flag\_Force  
     lrbm.h, 375  
**LBM\_SRC\_EVENT\_UME\_-**  
     Message\_Stable  
     lrbm.h, 375  
**LBM\_SRC\_EVENT\_UME\_-**  
     Message\_Stable\_Ex  
     lrbm.h, 375  
**LBM\_SRC\_EVENT\_UME\_-**  
     Message\_Stable\_Ex\_Flag\_Intergroup  
     Stable  
     lrbm.h, 375  
**LBM\_SRC\_EVENT\_UME\_-**  
     Message\_Stable\_Ex\_Flag\_Intragroup  
     Stable  
     lrbm.h, 375  
**LBM\_SRC\_EVENT\_UME\_-**  
     Message\_Stable\_Ex\_Flag\_Stable  
     lrbm.h, 375  
**LBM\_SRC\_EVENT\_UME\_-**  
     Message\_Stable\_Ex\_Flag\_Stable  
     lrbm.h, 375  
**LBM\_SRC\_EVENT\_UME\_-**  
     Message\_Stable\_Ex\_Flag\_Store  
     lrbm.h, 375  
**LBM\_SRC\_EVENT\_UME\_-**  
     Message\_Stable\_Ex\_Flag\_User  
     lrbm.h, 376

**LBM\_SRC\_EVENT\_UME\_-**  
     Message\_Stable\_Ex\_Flag\_Whole\_Message  
     Stable  
     lrbm.h, 376  
**LBM\_SRC\_EVENT\_UME\_-**  
     Registration\_Ex  
     Complete\_Ex  
     lrbm.h, 376  
**LBM\_SRC\_EVENT\_UME\_-**  
     Registration\_Ex\_Flag\_Qorum  
     lrbm.h, 376  
 lrbm\_src\_event\_ume\_registration\_-  
     complete\_ex\_t  
     lrbm.h, 399  
 lrbm\_src\_event\_ume\_registration\_-  
     complete\_ex\_t\_stct, 206  
     flags, 206  
     sequence\_number, 206  
**LBM\_SRC\_EVENT\_UME\_-**  
     Registration\_Error  
     lrbm.h, 376  
 lrbm\_src\_event\_ume\_registration\_ex\_t  
     lrbm.h, 400  
 lrbm\_src\_event\_ume\_registration\_ex\_t\_-  
     stct, 207  
     flags, 207  
     registration\_id, 207  
     sequence\_number, 207  
     store, 207  
     store\_index, 207  
**LBM\_SRC\_EVENT\_UME\_-**  
     Registration\_Success  
     lrbm.h, 376  
**LBM\_SRC\_EVENT\_UME\_-**  
     Registration\_Ex  
     Success\_Ex  
     lrbm.h, 376  
**LBM\_SRC\_EVENT\_UME\_-**  
     Registration\_Ex\_Flag\_Noacks  
     lrbm.h, 376

LBM\_SRC\_EVENT\_UME\_-  
REGISTRATION\_-  
SUCCESS\_EX\_FLAG\_OLD  
lrbm.h, 377

LBM\_SRC\_EVENT\_UME\_-  
REGISTRATION\_-  
SUCCESS\_EX\_FLAG\_RPP  
lrbm.h, 377

lrbm\_src\_event\_ume\_registration\_t  
lrbm.h, 400

lrbm\_src\_event\_ume\_registration\_t\_stct,  
209  
registration\_id, 209

LBM\_SRC\_EVENT\_UME\_STORE\_-  
UNRESPONSIVE  
lrbm.h, 377

LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_ID\_INFO  
lrbm.h, 377

lrbm\_src\_event\_umq\_message\_id\_info\_t  
lrbm.h, 400

lrbm\_src\_event\_umq\_message\_id\_info\_t\_stct, 210  
flags, 210  
msg\_clientd, 210  
msg\_id, 210

LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_STABLE\_EX  
lrbm.h, 377

LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_INTEGRGROU\_-  
STABLE  
lrbm.h, 377

LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_INTRAGROUP\_-  
STABLE  
lrbm.h, 377

LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_STABLE  
lrbm.h, 378

LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_USER  
lrbm.h, 378

lrbm.h, 378

LBM\_SRC\_EVENT\_UMQ\_-  
REGISTRATION\_-  
COMPLETE\_EX  
lrbm.h, 378

LBM\_SRC\_EVENT\_UMQ\_-  
REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_-  
QUORUM  
lrbm.h, 378

lrbm\_src\_event\_umq\_registration\_-  
complete\_ex\_t  
lrbm.h, 400

lrbm\_src\_event\_umq\_registration\_-  
complete\_ex\_t\_stct, 212  
flags, 212  
queue, 212  
queue\_id, 212

LBM\_SRC\_EVENT\_UMQ\_-  
REGISTRATION\_ERROR  
lrbm.h, 378

lrbm\_src\_event\_umq\_stability\_ack\_info\_-  
ex\_t  
lrbm.h, 400

lrbm\_src\_event\_umq\_stability\_ack\_info\_-  
ex\_t\_stct, 213  
first\_sequence\_number, 213  
flags, 213  
last\_sequence\_number, 214  
msg\_clientd, 214  
msg\_id, 213  
queue, 214  
queue\_id, 214  
queue\_instance, 214  
queue\_instance\_index, 214

LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_ASSIGNED\_EX  
lrbm.h, 378

LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_COMPLETE\_EX  
lrbm.h, 378

LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_CONSUMED\_EX  
lrbm.h, 379

lrbm\_src\_event\_umq\_ulb\_message\_info\_-  
ex\_t

lbm.h, 400  
 lbm\_src\_event\_umq\_ulb\_message\_info\_-  
     ex\_t\_stct, 215  
     application\_set\_index, 216  
     assignment\_id, 216  
     first\_sequence\_number, 216  
     flags, 215  
     last\_sequence\_number, 216  
     msg\_clientd, 216  
     msg\_id, 215  
     receiver, 216  
     registration\_id, 215  
 LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
     MESSAGE\_REASSIGNED\_-  
         EX  
         lbm.h, 379  
 LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
     MESSAGE\_REASSIGNED\_-  
         EX\_FLAG\_EXPLICIT  
         lbm.h, 379  
 LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
     MESSAGE\_TIMEOUT\_EX  
         lbm.h, 379  
 LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
     MESSAGE\_TIMEOUT\_EX\_-  
         FLAG\_DISCARD  
         lbm.h, 379  
 LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
     MESSAGE\_TIMEOUT\_EX\_-  
         FLAG\_EXPLICIT  
         lbm.h, 379  
 LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
     MESSAGE\_TIMEOUT\_EX\_-  
         FLAG\_MAX\_REASSIGNS  
         lbm.h, 379  
 LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
     MESSAGE\_TIMEOUT\_EX\_-  
         FLAG\_TOTAL\_LIFETIME\_-  
             EXPIRED  
             lbm.h, 380  
 LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
     ULB\_RECEIVER\_-  
         DEREGISTRATION\_EX  
         lbm.h, 380  
 lbm\_src\_event\_umq\_ulb\_receiver\_info\_-  
     ex\_t

lbm.h, 400  
 lbm\_src\_event\_umq\_ulb\_receiver\_info\_-  
     ex\_t\_stct, 217  
     application\_set\_index, 217  
     assignment\_id, 217  
     flags, 217  
     receiver, 217  
     registration\_id, 217  
 LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
     RECEIVER\_READY\_EX  
     lbm.h, 380  
 LBM\_SRC\_EVENT\_UMQ\_-  
     ULB\_RECEIVER\_-  
     REGISTRATION\_EX  
     lbm.h, 380  
 LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
     RECEIVER\_TIMEOUT\_EX  
     lbm.h, 380  
 LBM\_SRC\_EVENT\_WAKEUP  
     lbm.h, 380  
 LBM\_SRC\_EVENT\_WAKEUP\_-  
     FLAG\_MIM  
     lbm.h, 380  
 LBM\_SRC\_EVENT\_WAKEUP\_-  
     FLAG\_NORMAL  
     lbm.h, 381  
 LBM\_SRC\_EVENT\_WAKEUP\_-  
     FLAG\_REQUEST  
     lbm.h, 381  
 LBM\_SRC\_EVENT\_WAKEUP\_-  
     FLAG\_RESPONSE  
     lbm.h, 381  
 LBM\_SRC\_EVENT\_WAKEUP\_-  
     FLAG\_UIM  
     lbm.h, 381  
 lbm\_src\_event\_wakeup\_t  
     lbm.h, 401  
 lbm\_src\_event\_wakeup\_t\_stct, 219  
     flags, 219  
 lbm\_src\_flush  
     lbm.h, 503  
 lbm\_src\_get\_inflight  
     lbm.h, 503  
 lbm\_src\_get\_inflight\_ex  
     lbm.h, 504  
 lbm\_src\_getopt

lbtm.h, 504  
LBM\_SRC\_NONBLOCK  
    lbtm.h, 381  
lbtm\_src\_notify\_func\_t  
    lbtm.h, 401  
lbtm\_src\_notify\_func\_t\_stct, 220  
lbtm\_src\_notify\_function\_cb  
    lbtm.h, 401  
lbtm\_src\_reset\_transport\_stats  
    lbtm.h, 505  
lbtm\_src\_retrieve\_transport\_stats  
    lbtm.h, 505  
lbtm\_src\_send  
    lbtm.h, 505  
lbtm\_src\_send\_ex  
    lbtm.h, 506  
LBM\_SRC\_SEND\_EX\_FLAG\_-  
    APPHDR\_CHAIN  
        lbtm.h, 381  
LBM\_SRC\_SEND\_EX\_FLAG\_-  
    CHANNEL  
        lbtm.h, 381  
LBM\_SRC\_SEND\_EX\_FLAG\_HF\_32  
    lbtm.h, 381  
LBM\_SRC\_SEND\_EX\_FLAG\_HF\_64  
    lbtm.h, 381  
LBM\_SRC\_SEND\_EX\_FLAG\_HF\_-  
    OPTIONAL  
        lbtm.h, 382  
LBM\_SRC\_SEND\_EX\_FLAG\_-  
    PROPERTIES  
        lbtm.h, 382  
LBM\_SRC\_SEND\_EX\_FLAG\_-  
    SEQUENCE\_NUMBER\_-  
        INFO  
            lbtm.h, 382  
LBM\_SRC\_SEND\_EX\_FLAG\_-  
    SEQUENCE\_NUMBER\_-  
        INFO\_FRAGONLY  
            lbtm.h, 382  
LBM\_SRC\_SEND\_EX\_FLAG\_UME\_-  
    CLIENTD  
        lbtm.h, 382  
LBM\_SRC\_SEND\_EX\_FLAG\_UMQ\_-  
    CLIENTD  
        lbtm.h, 382

LBM\_SRC\_SEND\_EX\_FLAG\_UMQ\_-  
    INDEX  
        lbtm.h, 382  
LBM\_SRC\_SEND\_EX\_FLAG\_UMQ\_-  
    MESSAGE\_ID\_INFO  
        lbtm.h, 382  
LBM\_SRC\_SEND\_EX\_FLAG\_UMQ\_-  
    TOTAL\_LIFETIME  
        lbtm.h, 382  
lbtm\_src\_send\_ex\_info\_t  
    lbtm.h, 401  
lbtm\_src\_send\_ex\_info\_t\_stct, 221  
    apphdr\_chain, 222  
    async\_opfunc, 222  
    channel\_info, 221  
    flags, 221  
    hf\_sqn, 222  
    properties, 222  
    ume\_msg\_clientd, 221  
    umq\_index, 222  
    umq\_total\_lifetime, 222  
lbtm\_src\_sendv  
    lbtm.h, 507  
lbtm\_src\_sendv\_ex  
    lbtm.h, 508  
lbtm\_src\_setopt  
    lbtm.h, 509  
lbtm\_src\_str\_getopt  
    lbtm.h, 509  
lbtm\_src\_str\_setopt  
    lbtm.h, 509  
lbtm\_src\_topic\_alloc  
    lbtm.h, 510  
lbtm\_src\_topic\_attr\_create  
    lbtm.h, 510  
lbtm\_src\_topic\_attr\_create\_default  
    lbtm.h, 511  
lbtm\_src\_topic\_attr\_create\_from\_xml  
    lbtm.h, 511  
lbtm\_src\_topic\_attr\_delete  
    lbtm.h, 512  
lbtm\_src\_topic\_attr\_dump  
    lbtm.h, 512  
lbtm\_src\_topic\_attr\_dup  
    lbtm.h, 512  
lbtm\_src\_topic\_attr\_getopt

lbm.h, 512  
 lbm\_src\_topic\_attr\_option\_size  
     lbm.h, 513  
 lbm\_src\_topic\_attr\_set\_from\_xml  
     lbm.h, 513  
 lbm\_src\_topic\_attr\_setopt  
     lbm.h, 513  
 lbm\_src\_topic\_attr\_str\_getopt  
     lbm.h, 514  
 lbm\_src\_topic\_attr\_str\_setopt  
     lbm.h, 514  
 lbm\_src\_topic\_dump  
     lbm.h, 515  
 lbm\_src\_transport\_stats\_daemon\_t  
     lbm.h, 401  
 lbm\_src\_transport\_stats\_daemon\_t\_stct,  
     223  
     bytes\_buffered, 223  
 lbm\_src\_transport\_stats\_lbtipc\_t\_stct,  
     224  
     bytes\_sent, 224  
     msgs\_sent, 224  
     num\_clients, 224  
 lbm\_src\_transport\_stats\_lbtrdma\_t\_stct,  
     225  
     bytes\_sent, 225  
     msgs\_sent, 225  
     num\_clients, 225  
 lbm\_src\_transport\_stats\_lbtrm\_t\_stct,  
     226  
     bytes\_sent, 226  
     msgs\_sent, 226  
     nak\_pkts\_rcved, 227  
     naks\_ignored, 227  
     naks\_rcved, 227  
     naks\_rx\_delay\_ignored, 227  
     naks\_shed, 227  
     rctlr\_data\_msgs, 228  
     rctlr\_rx\_msgs, 228  
     rx\_bytes\_sent, 228  
     rxs\_sent, 228  
     txw\_bytes, 227  
     txw\_msgs, 226  
 lbm\_src\_transport\_stats\_lbtru\_t\_stct, 230  
     bytes\_sent, 230  
     msgs\_sent, 230  
     nak\_pkts\_rcved, 230  
     naks\_ignored, 231  
     naks\_rcved, 230  
     naks\_rx\_delay\_ignored, 231  
     naks\_shed, 231  
     num\_clients, 232  
     rx\_bytes\_sent, 232  
     rxs\_sent, 231  
 lbm\_src\_transport\_stats\_t  
     lbm.h, 402  
 lbm\_src\_transport\_stats\_t\_stct, 233  
     daemon, 234  
     lbtipc, 234  
     lbtrdma, 234  
     lbtrm, 234  
     lbtru, 234  
     source, 233  
     tcp, 234  
     type, 233  
 lbm\_src\_transport\_stats\_tcp\_t\_stct, 235  
     bytes\_buffered, 235  
     num\_clients, 235  
 lbm\_src\_ume\_deregister  
     lbm.h, 515  
 lbm\_str\_hash\_func\_ex\_t  
     lbm.h, 402  
 lbm\_str\_hash\_func\_ex\_t\_stct, 236  
     hashfunc, 236  
 lbm\_str\_hash\_function\_cb  
     lbm.h, 402  
 lbm\_str\_hash\_function\_cb\_ex  
     lbm.h, 402  
 lbm\_timer\_cb\_proc  
     lbm.h, 403  
 lbm\_timeval\_t  
     lbm.h, 403  
 lbm\_timeval\_t\_stct, 237  
 lbm\_topic\_from\_src  
     lbm.h, 515  
 LBM\_TOPIC\_RES\_REQUEST\_-  
     ADVERTISEMENT  
     lbm.h, 383  
 LBM\_TOPIC\_RES\_REQUEST\_QUERY  
     lbm.h, 383  
 LBM\_TOPIC\_RES\_REQUEST\_-  
     RESERVED1

lbt.h, 383  
LBM\_TOPIC\_RES\_REQUEST\_-  
    WILDCARD\_QUERY  
        lbt.h, 383  
lbt\_transport\_source\_format  
    lbt.h, 515  
lbt\_transport\_source\_info\_t  
    lbt.h, 403  
lbt\_transport\_source\_info\_t\_stct, 238  
    dest\_port, 239  
    mc\_group, 239  
    session\_id, 239  
    src\_ip, 239  
    src\_port, 239  
    topic\_idx, 239  
    transport\_id, 239  
    type, 239  
lbt\_transport\_source\_parse  
    lbt.h, 516  
LBM\_TRANSPORT\_STAT\_DAEMON  
    lbt.h, 383  
LBM\_TRANSPORT\_STAT\_LBTIPC  
    lbt.h, 383  
LBM\_TRANSPORT\_STAT\_LBTRDMA  
    lbt.h, 383  
LBM\_TRANSPORT\_STAT\_LBTRM  
    lbt.h, 383  
LBM\_TRANSPORT\_STAT\_LBTRU  
    lbt.h, 384  
LBM\_TRANSPORT\_STAT\_TCP  
    lbt.h, 384  
LBM\_TRANSPORT\_TYPE\_LBTIPC  
    lbt.h, 384  
LBM\_TRANSPORT\_TYPE\_-  
    LBTRDMA  
    lbt.h, 384  
LBM\_TRANSPORT\_TYPE\_LBTRM  
    lbt.h, 384  
LBM\_TRANSPORT\_TYPE\_LBTRU  
    lbt.h, 384  
LBM\_TRANSPORT\_TYPE\_TCP  
    lbt.h, 384  
lbt\_icast\_resolver\_entry\_t  
    lbt.h, 404  
lbt\_icast\_resolver\_entry\_t\_stct, 241  
    destination\_port, 241  
    iface, 241  
    resolver\_ip, 241  
    source\_port, 241  
lbt\_ume\_ack\_delete  
    lbt.h, 516  
lbt\_ume\_ack\_send\_explicit\_ack  
    lbt.h, 516  
lbt\_ume\_ctx\_rcv\_ctx\_notification\_-  
    create\_function\_cb  
    lbt.h, 404  
lbt\_ume\_ctx\_rcv\_ctx\_notification\_-  
    delete\_function\_cb  
    lbt.h, 404  
lbt\_ume\_ctx\_rcv\_ctx\_notification\_-  
    func\_t  
    lbt.h, 405  
lbt\_ume\_ctx\_rcv\_ctx\_notification\_-  
    func\_t\_stct, 243  
lbt\_ume\_rcv\_recovery\_info\_ex\_func\_-  
    info\_t  
    lbt.h, 405  
lbt\_ume\_rcv\_recovery\_info\_ex\_func\_-  
    info\_t\_stct, 244  
    flags, 244  
    high\_sequence\_number, 245  
    low\_rxreq\_max\_sequence\_number,  
        244  
    low\_sequence\_number, 244  
    source, 245  
    source\_clientd, 245  
lbt\_ume\_rcv\_recovery\_info\_ex\_func\_t  
    lbt.h, 405  
lbt\_ume\_rcv\_recovery\_info\_ex\_func\_-  
    t\_stct, 246  
lbt\_ume\_rcv\_recovery\_info\_ex\_-  
    function\_cb  
    lbt.h, 405  
lbt\_ume\_rcv\_regid\_ex\_func\_info\_t  
    lbt.h, 406  
lbt\_ume\_rcv\_regid\_ex\_func\_info\_t\_-  
    stct, 247  
    flags, 247  
    source, 248  
    source\_clientd, 247  
    src\_registration\_id, 247  
    store, 248

store\_index, 247  
**lbt\_ume\_rcv\_regid\_ex\_func\_t**  
 lbm.h, 406  
**lbt\_ume\_rcv\_regid\_ex\_func\_t\_stct**, 249  
**lbt\_ume\_rcv\_regid\_ex\_function\_cb**  
 lbm.h, 406  
**lbt\_ume\_rcv\_regid\_func\_t**  
 lbm.h, 407  
**lbt\_ume\_rcv\_regid\_func\_t\_stct**, 250  
**lbt\_ume\_rcv\_regid\_function\_cb**  
 lbm.h, 407  
**lbt\_ume\_src\_force\_reclaim\_func\_t**  
 lbm.h, 407  
**lbt\_ume\_src\_force\_reclaim\_func\_t\_stct**,  
 251  
**lbt\_ume\_src\_force\_reclaim\_function\_cb**  
 lbm.h, 407  
**lbt\_ume\_src\_msg\_stable**  
 lbm.h, 517  
**lbt\_ume\_store\_entry\_t**  
 lbm.h, 408  
**lbt\_ume\_store\_entry\_t\_stct**, 252  
 group\_index, 252  
 ip\_address, 252  
 registration\_id, 252  
 tcp\_port, 252  
**lbt\_ume\_store\_group\_entry\_t**  
 lbm.h, 408  
**lbt\_ume\_store\_group\_entry\_t\_stct**, 253  
 group\_size, 253  
 index, 253  
**lbt\_ume\_store\_name\_entry\_t**  
 lbm.h, 408  
**lbt\_ume\_store\_name\_entry\_t\_stct**, 254  
 group\_index, 254  
 name, 254  
 registration\_id, 254  
**LBM\_UMM\_INFO\_FLAGS\_USE\_SSL**  
 lbm.h, 384  
**lbt\_umm\_info\_t\_stct**, 255  
 appname, 255  
 cert\_file, 256  
 cert\_file\_password, 256  
 flags, 255  
 password, 255  
 servers, 255  
 username, 255  
**lbt\_umq\_ctx\_msg\_stable**  
 lbm.h, 517  
**LBM\_UMQ\_INDEX\_FLAG\_NUMERIC**  
 lbm.h, 384  
**lbt\_umq\_index\_info\_t**  
 lbm.h, 408  
**lbt\_umq\_index\_info\_t\_stct**, 257  
 flags, 257  
 index, 257  
 index\_len, 257  
**lbt\_umq\_msg\_selector\_create**  
 lbm.h, 517  
**lbt\_umq\_msg\_selector\_delete**  
 lbm.h, 518  
**lbt\_umq\_msg\_total\_lifetime\_info\_t**  
 lbm.h, 408  
**lbt\_umq\_msg\_total\_lifetime\_info\_t\_stct**,  
 258  
 flags, 258  
 umq\_msg\_total\_lifetime, 258  
**lbt\_umq\_msqid\_t**  
 lbm.h, 409  
**lbt\_umq\_msqid\_t\_stct**, 259  
 regid, 259  
 stamp, 259  
**lbt\_umq\_queue\_entry\_t**  
 lbm.h, 409  
**lbt\_umq\_queue\_entry\_t\_stct**, 260  
 name, 260  
 regid, 260  
**lbt\_umq\_queue\_msg\_status\_t**, 261  
 clientd, 261  
 flags, 262  
 msg, 261  
 msgid, 261  
 status, 261  
**lbt\_umq\_queue\_topic\_status\_t**, 263  
 flags, 263  
 status, 263  
 topic, 263  
**lbt\_umq\_queue\_topic\_t\_stct**, 264  
 appsets, 264  
 num\_appsets, 264

reserved, 264  
topic\_name, 264

lbm\_umq\_regid\_t  
  lbm.h, 409

lbm\_umq\_ulb\_application\_set\_attr\_t  
  lbm.h, 409

lbm\_umq\_ulb\_application\_set\_attr\_t\_stct, 265  
  d, 265  
  index, 265  
  lu, 265  
  value, 265

lbm\_umq\_ulb\_receiver\_type\_attr\_t  
  lbm.h, 409

lbm\_umq\_ulb\_receiver\_type\_attr\_t\_stct, 266  
  d, 266  
  id, 266  
  lu, 266  
  value, 266

lbm\_umq\_ulb\_receiver\_type\_entry\_t  
  lbm.h, 409

lbm\_umq\_ulb\_receiver\_type\_entry\_t\_stct, 267  
  application\_set\_index, 267  
  id, 267

lbm\_unicast\_immediate\_message  
  lbm.h, 518

lbm\_unicast\_immediate\_request  
  lbm.h, 518

lbm\_version  
  lbm.h, 519

lbm\_wildcard\_rcv\_attr\_create  
  lbm.h, 519

lbm\_wildcard\_rcv\_attr\_create\_default  
  lbm.h, 520

lbm\_wildcard\_rcv\_attr\_create\_from\_xml  
  lbm.h, 520

lbm\_wildcard\_rcv\_attr\_delete  
  lbm.h, 521

lbm\_wildcard\_rcv\_attr\_dump  
  lbm.h, 521

lbm\_wildcard\_rcv\_attr\_dup  
  lbm.h, 521

lbm\_wildcard\_rcv\_attr\_getopt  
  lbm.h, 522

lbm\_wildcard\_rcv\_attr\_option\_size  
  lbm.h, 522

lbm\_wildcard\_rcv\_attr\_set\_from\_xml  
  lbm.h, 522

lbm\_wildcard\_rcv\_attr\_setopt  
  lbm.h, 523

lbm\_wildcard\_rcv\_attr\_str\_getopt  
  lbm.h, 523

lbm\_wildcard\_rcv\_attr\_str\_setopt  
  lbm.h, 523

lbm\_wildcard\_rcv\_COMPARE\_func\_t  
  lbm.h, 409

lbm\_wildcard\_rcv\_COMPARE\_func\_t\_stct, 268

lbm\_wildcard\_rcv\_COMPARE\_function\_cb  
  lbm.h, 409

lbm\_wildcard\_rcv\_create  
  lbm.h, 524

lbm\_wildcard\_rcv\_create\_func\_t  
  lbm.h, 410

lbm\_wildcard\_rcv\_create\_func\_t\_stct, 269

lbm\_wildcard\_rcv\_create\_function\_cb  
  lbm.h, 410

lbm\_wildcard\_rcv\_delete  
  lbm.h, 525

lbm\_wildcard\_rcv\_delete\_ex  
  lbm.h, 525

lbm\_wildcard\_rcv\_delete\_func\_t  
  lbm.h, 411

lbm\_wildcard\_rcv\_delete\_func\_t\_stct, 270

lbm\_wildcard\_rcv\_DELETE\_function\_cb  
  lbm.h, 411

lbm\_wildcard\_rcv\_DUMP  
  lbm.h, 525

lbm\_wildcard\_rcv\_getopt  
  lbm.h, 526

LBM\_WILDCARD\_RCV\_PATTERN\_TYPE\_APP\_CB  
  lbm.h, 385

LBM\_WILDCARD\_RCV\_PATTERN\_TYPE\_PCRE  
  lbm.h, 385

LBM\_WILDCARD\_RCV\_PATTERN\_TYPE\_REGEX

lbm.h, 385  
 lbm\_wildcard\_rcv\_setopt  
     lbm.h, 526  
 lbm\_wildcard\_rcv\_str\_getopt  
     lbm.h, 526  
 lbm\_wildcard\_rcv\_str\_setopt  
     lbm.h, 527  
 lbm\_wildcard\_rcv\_subscribe\_channel  
     lbm.h, 527  
 lbm\_wildcard\_rcv\_umq\_deregister  
     lbm.h, 528  
 lbm\_wildcard\_rcv\_umq\_index\_release  
     lbm.h, 528  
 lbm\_wildcard\_rcv\_umq\_index\_start\_assignment  
     lbm.h, 528  
 lbm\_wildcard\_rcv\_umq\_index\_stop\_assignment  
     lbm.h, 529  
 lbm\_wildcard\_rcv\_unsubscribe\_channel  
     lbm.h, 529  
 lbm\_wildcard\_rcv\_unsubscribe\_channel\_ex  
     lbm.h, 529  
 lbm\_win32\_static\_thread\_attach  
     lbm.h, 530  
 lbm\_win32\_static\_thread\_detach  
     lbm.h, 530  
 lbm\_wrcv\_ume\_deregister  
     lbm.h, 530  
 lbmaux.h, 532  
     lbmaux\_context\_attr\_setopt\_from\_file, 533  
     lbmaux\_context\_create\_from\_file, 534  
     lbmaux\_event\_queue\_attr\_setopt\_from\_file, 534  
     lbmaux\_rcv\_topic\_attr\_setopt\_from\_file, 534  
     lbmaux\_src\_topic\_attr\_setopt\_from\_file, 535  
     lbmaux\_wildcard\_rcv\_attr\_setopt\_from\_file, 535  
 lbmaux\_context\_attr\_setopt\_from\_file  
     lbmaux.h, 533  
 lbmaux\_context\_create\_from\_file  
     lbmaux.h, 534  
     lbmaux.event\_queue\_attr\_setopt\_from\_file  
         lbmaux.h, 534  
 lbmaux\_rcv\_topic\_attr\_setopt\_from\_file  
     lbmaux.h, 534  
 lbmaux\_src\_topic\_attr\_setopt\_from\_file  
     lbmaux.h, 535  
 lbmaux\_wildcard\_rcv\_attr\_setopt\_from\_file  
     lbmaux.h, 535  
 lbmht.h, 536  
     lbm\_delete\_cb\_proc, 539  
     lbm\_hypertopic\_rcv\_add, 540  
     lbm\_hypertopic\_rcv\_cb\_proc, 539  
     lbm\_hypertopic\_rcv\_delete, 540  
     lbm\_hypertopic\_rcv\_destroy, 540  
     lbm\_hypertopic\_rcv\_init, 541  
 lbmmmon.h, 542  
     LBMMON\_ATTR\_APPSOURCEID, 550  
     LBMMON\_ATTR\_CONTEXTID, 550  
     LBMMON\_ATTR\_FORMAT\_MODULEID, 550  
     lbmmmon\_attr\_get\_appsorceid, 563  
     lbmmmon\_attr\_get\_contextid, 563  
     lbmmmon\_attr\_get\_ip4sender, 564  
     lbmmmon\_attr\_get\_objectid, 564  
     lbmmmon\_attr\_get\_processid, 564  
     lbmmmon\_attr\_get\_source, 564  
     lbmmmon\_attr\_get\_timestamp, 565  
     LBMMON\_ATTR\_OBJECTID, 550  
     LBMMON\_ATTR\_PROCESSID, 550  
     LBMMON\_ATTR\_SENDER\_IPV4, 551  
     LBMMON\_ATTR\_SOURCE, 551  
     LBMMON\_ATTR\_SOURCE\_IM, 551  
     LBMMON\_ATTR\_SOURCE\_NORMAL, 551  
     LBMMON\_ATTR\_TIMESTAMP, 551  
     lbmmmon\_context\_monitor, 565  
     lbmmmon\_context\_unmonitor, 566

lbmmmon\_ctx\_format\_deserialize\_t,  
    553  
lbmmmon\_ctx\_format\_serialize\_t,  
    553  
lbmmmon\_ctx\_statistics\_cb, 554  
lbmmmon\_ctx\_statistics\_func\_t, 554  
LBMMON\_EAGAIN, 551  
LBMMON\_EALREADY, 551  
LBMMON\_EINVAL, 551  
LBMMON\_ELBMFAIL, 551  
LBMMON\_EMODFAIL, 552  
LBMMON\_ENOMEM, 552  
lbmmmon\_errmsg, 566  
lbmmmon\_errnum, 566  
LBMMON\_ERROR\_BASE, 552  
lbmmmon\_evq\_format\_deserialize\_t,  
    554  
lbmmmon\_evq\_format\_serialize\_t,  
    555  
lbmmmon\_evq\_monitor, 566  
lbmmmon\_evq\_statistics\_cb, 556  
lbmmmon\_evq\_statistics\_func\_t, 556  
lbmmmon\_evq\_unmonitor, 567  
lbmmmon\_format\_errmsg\_t, 556  
lbmmmon\_format\_finish\_t, 556  
lbmmmon\_format\_init\_t, 557  
lbmmmon\_next\_key\_value\_pair, 567  
lbmmmon\_packet\_hdr\_t, 557  
LBMMON\_PACKET\_-  
    SIGNATURE, 552  
LBMMON\_PACKET\_TYPE\_-  
    CONTEXT, 552  
LBMMON\_PACKET\_TYPE\_-  
    EVENT\_QUEUE, 552  
LBMMON\_PACKET\_TYPE\_-  
    RECEIVER, 552  
LBMMON\_PACKET\_TYPE\_-  
    SOURCE, 552  
lbmmmon\_rctl\_attr\_create, 568  
lbmmmon\_rctl\_attr\_delete, 568  
lbmmmon\_rctl\_attr\_getopt, 568  
lbmmmon\_rctl\_attr\_setopt, 569  
LBMMON\_RCTL\_CONTEXT\_-  
    CALLBACK, 552  
lbmmmon\_rctl\_create, 569  
lbmmmon\_rctl\_destroy, 570  
LBMMON\_RCTL\_EVENT\_-  
    QUEUE\_CALLBACK, 552  
LBMMON\_RCTL\_RECEIVER\_-  
    CALLBACK, 553  
LBMMON\_RCTL\_SOURCE\_-  
    CALLBACK, 553  
lbmmmon\_rcv\_format\_deserialize\_t,  
    557  
lbmmmon\_rcv\_format\_serialize\_t,  
    558  
lbmmmon\_rcv\_monitor, 570  
lbmmmon\_rcv\_statistics\_cb, 559  
lbmmmon\_rcv\_statistics\_func\_t, 559  
lbmmmon\_rcv\_unmonitor, 571  
lbmmmon\_sctl\_create, 571  
lbmmmon\_sctl\_destroy, 572  
lbmmmon\_sctl\_sample, 572  
lbmmmon\_sctl\_sample\_ex, 572  
lbmmmon\_src\_format\_deserialize\_t,  
    559  
lbmmmon\_src\_format\_serialize\_t, 560  
lbmmmon\_src\_monitor, 572  
lbmmmon\_src\_statistics\_cb, 560  
lbmmmon\_src\_statistics\_func\_t, 561  
lbmmmon\_src\_unmonitor, 573  
lbmmmon\_transport\_errmsg\_t, 561  
lbmmmon\_transport\_finishrecv\_t, 561  
lbmmmon\_transport\_finishsrc\_t, 561  
lbmmmon\_transport\_initrecv\_t, 562  
lbmmmon\_transport\_initsrc\_t, 562  
lbmmmon\_transport\_receive\_t, 562  
lbmmmon\_transport\_send\_t, 563  
LBMMON\_ATTR\_APPSOURCEID  
    lbmmmon.h, 550  
lbmmmon\_attr\_block\_t\_stct, 271  
    mEntryCount, 271  
    mEntryLength, 271  
LBMMON\_ATTR\_CONTEXTID  
    lbmmmon.h, 550  
lbmmmon\_attr\_entry\_t\_stct, 272  
    mLength, 272  
    mType, 272  
LBMMON\_ATTR\_FORMAT\_-  
    MODULEID  
    lbmmmon.h, 550  
lbmmmon\_attr\_get\_appsuserid

lbmmon.h, 563  
 lbmmon\_attr\_get\_contextid  
     lbmmon.h, 563  
 lbmmon\_attr\_get\_ipv4sender  
     lbmmon.h, 564  
 lbmmon\_attr\_get\_objectid  
     lbmmon.h, 564  
 lbmmon\_attr\_get\_processid  
     lbmmon.h, 564  
 lbmmon\_attr\_get\_source  
     lbmmon.h, 564  
 lbmmon\_attr\_get\_timestamp  
     lbmmon.h, 565  
**LBMMON\_ATTR\_OBJECTID**  
     lbmmon.h, 550  
**LBMMON\_ATTR\_PROCESSID**  
     lbmmon.h, 550  
**LBMMON\_ATTR\_SENDER\_IPV4**  
     lbmmon.h, 551  
**LBMMON\_ATTR\_SOURCE**  
     lbmmon.h, 551  
**LBMMON\_ATTR\_SOURCE\_IM**  
     lbmmon.h, 551  
**LBMMON\_ATTR\_SOURCE\_-NORMAL**  
     lbmmon.h, 551  
**LBMMON\_ATTR\_TIMESTAMP**  
     lbmmon.h, 551  
 lbmmon\_context\_monitor  
     lbmmon.h, 565  
 lbmmon\_context\_unmonitor  
     lbmmon.h, 566  
 lbmmon\_ctx\_format\_deserialize\_t  
     lbmmon.h, 553  
 lbmmon\_ctx\_format\_serialize\_t  
     lbmmon.h, 553  
 lbmmon\_ctx\_statistics\_cb  
     lbmmon.h, 554  
 lbmmon\_ctx\_statistics\_func\_t  
     lbmmon.h, 554  
 lbmmon\_ctx\_statistics\_func\_t\_stct, 273  
     cbfunc, 273  
**LBMMON\_EAGAIN**  
     lbmmon.h, 551  
**LBMMON\_EALREADY**  
     lbmmon.h, 551

LBMMON\_EINVAL  
     lbmmon.h, 551  
**LBMMON\_ELBMFAIL**  
     lbmmon.h, 551  
**LBMMON\_EMODFAIL**  
     lbmmon.h, 552  
**LBMMON\_ENOMEM**  
     lbmmon.h, 552  
 lbmmon\_errmsg  
     lbmmon.h, 566  
 lbmmon\_errnum  
     lbmmon.h, 566  
**LBMMON\_ERROR\_BASE**  
     lbmmon.h, 552  
 lbmmon\_evq\_format\_deserialize\_t  
     lbmmon.h, 554  
 lbmmon\_evq\_format\_serialize\_t  
     lbmmon.h, 555  
 lbmmon\_evq\_monitor  
     lbmmon.h, 566  
 lbmmon\_evq\_statistics\_cb  
     lbmmon.h, 556  
 lbmmon\_evq\_statistics\_func\_t  
     lbmmon.h, 556  
 lbmmon\_evq\_statistics\_func\_t\_stct, 274  
     cbfunc, 274  
 lbmmon\_evq\_unmonitor  
     lbmmon.h, 567  
 lbmmon\_format\_errmsg\_t  
     lbmmon.h, 556  
 lbmmon\_format\_finish\_t  
     lbmmon.h, 556  
 lbmmon\_format\_func\_t\_stct, 275  
     mCtxDeserialize, 276  
     mCtxSerialize, 276  
     mErrorMessage, 276  
     mEvqDeserialize, 276  
     mEvqSerialize, 276  
     mFinish, 276  
     mInit, 275  
     mRcvDeserialize, 275  
     mRcvSerialize, 275  
     mSrcDeserialize, 275  
     mSrcSerialize, 275  
 lbmmon\_format\_init\_t  
     lbmmon.h, 557

Ibmmon\_next\_key\_value\_pair  
    Ibmmon.h, 567

Ibmmon\_packet\_hdr\_t  
    Ibmmon.h, 557

Ibmmon\_packet\_hdr\_t\_stct, 277

- mAttributeBlockLength, 277
- mDataLength, 277
- mFiller, 277
- mSignature, 277
- mType, 277

LBMMON\_PACKET\_SIGNATURE

- Ibmmon.h, 552

LBMMON\_PACKET\_TYPE\_-  
    CONTEXT

- Ibmmon.h, 552

LBMMON\_PACKET\_TYPE\_EVENT\_-  
    QUEUE

- Ibmmon.h, 552

LBMMON\_PACKET\_TYPE\_-  
    RECEIVER

- Ibmmon.h, 552

LBMMON\_PACKET\_TYPE\_SOURCE

- Ibmmon.h, 552

Ibmmon\_rctl\_attr\_create  
    Ibmmon.h, 568

Ibmmon\_rctl\_attr\_delete  
    Ibmmon.h, 568

Ibmmon\_rctl\_attr\_getopt  
    Ibmmon.h, 568

Ibmmon\_rctl\_attr\_setopt  
    Ibmmon.h, 569

LBMMON\_RCTL\_CONTEXT\_-  
    CALLBACK

- Ibmmon.h, 552

Ibmmon\_rctl\_create  
    Ibmmon.h, 569

Ibmmon\_rctl\_destroy  
    Ibmmon.h, 570

LBMMON\_RCTL\_EVENT\_QUEUE\_-  
    CALLBACK

- Ibmmon.h, 552

LBMMON\_RCTL\_RECEIVER\_-  
    CALLBACK

- Ibmmon.h, 553

LBMMON\_RCTL\_SOURCE\_-  
    CALLBACK

- Ibmmon.h, 553

Ibmmon.h, 553

Ibmmon\_rcv\_format\_deserialize\_t  
    Ibmmon.h, 557

Ibmmon\_rcv\_format\_serialize\_t  
    Ibmmon.h, 558

Ibmmon\_rcv\_monitor  
    Ibmmon.h, 570

Ibmmon\_rcv\_statistics\_cb  
    Ibmmon.h, 559

Ibmmon\_rcv\_statistics\_func\_t  
    Ibmmon.h, 559

Ibmmon\_rcv\_statistics\_func\_t\_stct, 279

- cbfunc, 279

Ibmmon\_rcv\_unmonitor  
    Ibmmon.h, 571

Ibmmon\_sctl\_create  
    Ibmmon.h, 571

Ibmmon\_sctl\_destroy  
    Ibmmon.h, 572

Ibmmon\_sctl\_sample  
    Ibmmon.h, 572

Ibmmon\_sctl\_sample\_ex  
    Ibmmon.h, 572

Ibmmon\_src\_format\_deserialize\_t  
    Ibmmon.h, 559

Ibmmon\_src\_format\_serialize\_t  
    Ibmmon.h, 560

Ibmmon\_src\_monitor  
    Ibmmon.h, 572

Ibmmon\_src\_statistics\_cb  
    Ibmmon.h, 560

Ibmmon\_src\_statistics\_func\_t  
    Ibmmon.h, 561

Ibmmon\_src\_statistics\_func\_t\_stct, 280

- cbfunc, 280

Ibmmon\_src\_unmonitor  
    Ibmmon.h, 573

Ibmmon\_transport\_errmsg\_t  
    Ibmmon.h, 561

Ibmmon\_transport\_finishrecv\_t  
    Ibmmon.h, 561

Ibmmon\_transport\_finishsrc\_t  
    Ibmmon.h, 561

Ibmmon\_transport\_func\_t\_stct, 281

- mErrorMessage, 282
- mFinishReceiver, 281

mFinishSource, 281  
 mInitReceiver, 281  
 mInitSource, 281  
 mReceive, 281  
 mSend, 281  
 lbmmon\_transport\_initrecv\_t  
     lbmmon.h, 562  
 lbmmon\_transport\_initsrc\_t  
     lbmmon.h, 562  
 lbmmon\_transport\_receive\_t  
     lbmmon.h, 562  
 lbmmon\_transport\_send\_t  
     lbmmon.h, 563  
 lbmpdm.h, 574  
     lbmpdm\_cache\_init, 596  
     lbmpdm\_cache\_struct\_add, 596  
     lbmpdm\_cache\_struct\_find, 597  
     lbmpdm\_cache\_struct\_find\_by\_version, 597  
     lbmpdm\_cache\_struct\_remove, 597  
     lbmpdm\_cache\_struct\_removeByVersion, 597  
     lbmpdm\_defn\_add\_field\_info\_by\_int\_name, 597  
     lbmpdm\_defn\_add\_field\_info\_by\_str\_name, 598  
     lbmpdm\_defn\_create, 598  
     lbmpdm\_defn\_delete, 599  
     lbmpdm\_defn\_deserialize, 599  
     lbmpdm\_defn\_finalize, 600  
     lbmpdm\_defn\_get\_field\_handle\_by\_int\_name, 600  
     lbmpdm\_defn\_get\_field\_handle\_by\_str\_name, 600  
     lbmpdm\_defn\_get\_field\_info\_int\_name, 600  
     lbmpdm\_defn\_get\_field\_info\_str\_name, 601  
     lbmpdm\_defn\_get\_field\_info\_type, 601  
     lbmpdm\_defn\_get\_field\_names\_type, 601  
     lbmpdm\_defn\_get\_id, 601  
     lbmpdm\_defn\_get\_length, 602  
     lbmpdm\_defn\_get\_msg\_vers\_major, 602  
         lbmpdm\_defn\_get\_msg\_vers\_minor, 602  
         lbmpdm\_defn\_get\_num\_fields, 602  
         lbmpdm\_defn\_is\_finalized, 603  
         lbmpdm\_defn\_serialize, 603  
         lbmpdm\_errmsg, 603  
         lbmpdm\_errnum, 603  
         lbmpdm\_field\_value\_stct\_delete, 603  
         lbmpdm\_iter\_create, 604  
         lbmpdm\_iter\_create\_from\_field\_handle, 604  
         lbmpdm\_iter\_delete, 604  
         lbmpdm\_iter\_first, 605  
         lbmpdm\_iter\_get\_current, 605  
         lbmpdm\_iter\_get\_current\_field\_value, 605  
         lbmpdm\_iter\_get\_current\_field\_value\_vec, 605  
         lbmpdm\_iter\_has\_next, 606  
         lbmpdm\_iter\_is\_current\_set, 606  
         lbmpdm\_iter\_next, 606  
         lbmpdm\_iter\_set\_current\_field\_value, 607  
         lbmpdm\_iter\_set\_current\_field\_value\_vec, 607  
         lbmpdm\_iter\_set\_msg, 607  
         lbmpdm\_msg\_and\_defn\_delete, 607  
         lbmpdm\_msg\_create, 608  
         lbmpdm\_msg\_delete, 608  
         lbmpdm\_msg\_deserialize, 608  
         lbmpdm\_msg\_get\_data, 609  
         lbmpdm\_msg\_get\_defn, 609  
         lbmpdm\_msg\_get\_field\_value, 609  
         lbmpdm\_msg\_get\_field\_value\_stct, 610  
         lbmpdm\_msg\_get\_field\_value\_vec, 610  
         lbmpdm\_msg\_get\_length, 611  
         lbmpdm\_msg\_is\_field\_set, 611  
         lbmpdm\_msg\_remove\_field\_value, 611  
         lbmpdm\_msg\_serialize, 611  
         lbmpdm\_msg\_set\_field\_value, 612  
         lbmpdm\_msg\_set\_field\_value\_vec, 612

lbmpdm\_msg\_set\_incl\_defn\_flag, 612  
lbmpdm\_msg\_unset\_incl\_defn\_flag, 613  
PDM\_DEFN\_INT\_FIELD\_-  
  NAMES, 589  
PDM\_DEFN\_STR\_FIELD\_-  
  NAMES, 589  
PDM\_ERR\_CREATE\_BUFFER,  
  589  
PDM\_ERR\_CREATE\_SECTION,  
  590  
PDM\_ERR\_DEFN\_INVALID, 590  
PDM\_ERR\_EINVAL, 590  
PDM\_ERR\_FIELD\_IS\_NULL, 590  
PDM\_ERR\_FIELD\_NOT\_FOUND,  
  590  
PDM\_ERR\_INSUFFICIENT\_-  
  BUFFER\_LENGTH, 590  
PDM\_ERR\_MSG\_INVALID, 590  
PDM\_ERR\_NO\_MORE\_FIELDS,  
  590  
PDM\_ERR\_NOMEM, 590  
PDM\_ERR\_REQ\_FIELD\_NOT\_-  
  SET, 590  
PDM\_FAILURE, 591  
PDM\_FALSE, 591  
PDM\_FIELD\_INFO\_FLAG\_-  
  FIXED\_STR\_LEN, 591  
PDM\_FIELD\_INFO\_FLAG\_-  
  NUM\_ARR\_ELEM, 591  
PDM\_FIELD\_INFO\_FLAG\_REQ,  
  591  
PDM\_INTERNAL\_TYPE\_-  
  INVALID, 591  
PDM\_ITER\_INVALID\_FIELD\_-  
  HANDLE, 591  
PDM\_MSG\_FLAG\_DEL\_DEFN\_-  
  WHEN\_REPLACED, 591  
PDM\_MSG\_FLAG\_INCL\_DEFN,  
  591  
PDM\_MSG\_FLAG\_NEED\_-  
  BYTE\_SWAP, 592  
PDM\_MSG\_FLAG\_TRY\_LOAD\_-  
  DEFN\_FROM\_CACHE, 592  
PDM\_MSG\_FLAG\_USE\_MSG\_-  
  DEFN\_IF\_NEEDED, 592  
PDM\_MSG\_FLAG\_VAR\_OR\_-  
  OPT\_FLDS\_SET, 592  
PDM\_MSG\_VER\_POLICY\_BEST,  
  592  
PDM\_MSG\_VER\_POLICY\_-  
  EXACT, 592  
PDM\_SUCCESS, 592  
PDM\_TRUE, 592  
PDM\_TYPE\_BLOB, 592  
PDM\_TYPE\_BLOB\_ARR, 593  
PDM\_TYPE\_BOOLEAN, 593  
PDM\_TYPE\_BOOLEAN\_ARR,  
  593  
PDM\_TYPE\_DECIMAL, 593  
PDM\_TYPE\_DECIMAL\_ARR, 593  
PDM\_TYPE\_DOUBLE, 593  
PDM\_TYPE\_DOUBLE\_ARR, 593  
PDM\_TYPE\_FIX\_STRING, 593  
PDM\_TYPE\_FIX\_STRING\_ARR,  
  593  
PDM\_TYPE\_FIX\_UNICODE, 593  
PDM\_TYPE\_FIX\_UNICODE\_-  
  ARR, 594  
PDM\_TYPE\_FLOAT, 594  
PDM\_TYPE\_FLOAT\_ARR, 594  
PDM\_TYPE\_INT16, 594  
PDM\_TYPE\_INT16\_ARR, 594  
PDM\_TYPE\_INT32, 594  
PDM\_TYPE\_INT32\_ARR, 594  
PDM\_TYPE\_INT64, 594  
PDM\_TYPE\_INT64\_ARR, 594  
PDM\_TYPE\_INT8, 594  
PDM\_TYPE\_INT8\_ARR, 595  
PDM\_TYPE\_MESSAGE, 595  
PDM\_TYPE\_MESSAGE\_ARR,  
  595  
PDM\_TYPE\_STRING, 595  
PDM\_TYPE\_STRING\_ARR, 595  
PDM\_TYPE\_TIMESTAMP, 595  
PDM\_TYPE\_TIMESTAMP\_ARR,  
  595  
PDM\_TYPE\_UINT16, 595  
PDM\_TYPE\_UINT16\_ARR, 595  
PDM\_TYPE\_UINT32, 595

PDM\_TYPE\_UINT32\_ARR, 596  
 PDM\_TYPE\_UINT64, 596  
 PDM\_TYPE\_UINT64\_ARR, 596  
 PDM\_TYPE\_UINT8, 596  
 PDM\_TYPE\_UINT8\_ARR, 596  
 PDM\_TYPE\_UNICODE, 596  
 PDM\_TYPE\_UNICODE\_ARR, 596  
 lbmpdm\_cache\_init  
     lbmpdm.h, 596  
 lbmpdm\_cache\_struct\_add  
     lbmpdm.h, 596  
 lbmpdm\_cache\_struct\_find  
     lbmpdm.h, 597  
 lbmpdm\_cache\_struct\_find\_by\_version  
     lbmpdm.h, 597  
 lbmpdm\_cache\_struct\_remove  
     lbmpdm.h, 597  
 lbmpdm\_cache\_struct\_remove\_by\_version  
     lbmpdm.h, 597  
 lbmpdm\_decimal\_t, 283  
     exp, 283  
     mant, 283  
 lbmpdm\_defn\_add\_field\_info\_by\_int\_name  
     lbmpdm.h, 597  
 lbmpdm\_defn\_add\_field\_info\_by\_str\_name  
     lbmpdm.h, 598  
 lbmpdm\_defn\_create  
     lbmpdm.h, 598  
 lbmpdm\_defn\_delete  
     lbmpdm.h, 599  
 lbmpdm\_defn\_deserialize  
     lbmpdm.h, 599  
 lbmpdm\_defn\_finalize  
     lbmpdm.h, 600  
 lbmpdm\_defn\_get\_field\_handle\_by\_int\_name  
     lbmpdm.h, 600  
 lbmpdm\_defn\_get\_field\_handle\_by\_str\_name  
     lbmpdm.h, 600  
 lbmpdm\_defn\_get\_field\_info\_int\_name  
     lbmpdm.h, 600  
 lbmpdm\_defn\_get\_field\_info\_str\_name  
     lbmpdm.h, 601  
 lbmpdm\_defn\_get\_field\_info\_type  
     lbmpdm.h, 601  
 lbmpdm\_defn\_get\_field\_names\_type  
     lbmpdm.h, 601  
 lbmpdm\_defn\_get\_id  
     lbmpdm.h, 601  
 lbmpdm\_defn\_get\_length  
     lbmpdm.h, 602  
 lbmpdm\_defn\_get\_msg\_vers\_major  
     lbmpdm.h, 602  
 lbmpdm\_defn\_get\_msg\_vers\_minor  
     lbmpdm.h, 602  
 lbmpdm\_defn\_get\_num\_fields  
     lbmpdm.h, 602  
 lbmpdm\_defn\_is\_finalized  
     lbmpdm.h, 603  
 lbmpdm\_defn\_serialize  
     lbmpdm.h, 603  
 lbmpdm\_errmsg  
     lbmpdm.h, 603  
 lbmpdm\_errnum  
     lbmpdm.h, 603  
 lbmpdm\_field\_info\_attr\_stct\_t, 284  
     fixed\_str\_len, 284  
     num\_arr\_elem, 284  
     req, 284  
 lbmpdm\_field\_value\_stct\_delete  
     lbmpdm.h, 603  
 lbmpdm\_field\_value\_stct\_t, 285  
     field\_type, 285  
     is\_array, 285  
     is\_fixed, 285  
     len, 285  
     len\_arr, 286  
     num\_arr\_elem, 285  
     value, 285  
     value\_arr, 286  
 lbmpdm\_iter\_create  
     lbmpdm.h, 604  
 lbmpdm\_iter\_create\_from\_field\_handle  
     lbmpdm.h, 604  
 lbmpdm\_iter\_delete  
     lbmpdm.h, 604  
 lbmpdm\_iter\_first  
     lbmpdm.h, 605

lbmpdm\_iter\_get\_current  
  lbmpdm.h, 605  
lbmpdm\_iter\_get\_current\_field\_value  
  lbmpdm.h, 605  
lbmpdm\_iter\_get\_current\_field\_value\_-  
  vec  
  lbmpdm.h, 605  
lbmpdm\_iter\_has\_next  
  lbmpdm.h, 606  
lbmpdm\_iter\_is\_current\_set  
  lbmpdm.h, 606  
lbmpdm\_iter\_next  
  lbmpdm.h, 606  
lbmpdm\_iter\_set\_current\_field\_value  
  lbmpdm.h, 607  
lbmpdm\_iter\_set\_current\_field\_value\_-  
  vec  
  lbmpdm.h, 607  
lbmpdm\_iter\_set\_msg  
  lbmpdm.h, 607  
lbmpdm\_msg\_and\_defn\_delete  
  lbmpdm.h, 607  
lbmpdm\_msg\_create  
  lbmpdm.h, 608  
lbmpdm\_msg\_delete  
  lbmpdm.h, 608  
lbmpdm\_msg\_deserialize  
  lbmpdm.h, 608  
lbmpdm\_msg\_get\_data  
  lbmpdm.h, 609  
lbmpdm\_msg\_get\_defn  
  lbmpdm.h, 609  
lbmpdm\_msg\_get\_field\_value  
  lbmpdm.h, 609  
lbmpdm\_msg\_get\_field\_value\_stct  
  lbmpdm.h, 610  
lbmpdm\_msg\_get\_field\_value\_vec  
  lbmpdm.h, 610  
lbmpdm\_msg\_get\_length  
  lbmpdm.h, 611  
lbmpdm\_msg\_is\_field\_set  
  lbmpdm.h, 611  
lbmpdm\_msg\_remove\_field\_value  
  lbmpdm.h, 611  
lbmpdm\_msg\_serialize  
  lbmpdm.h, 611

lbmpdm\_msg\_set\_field\_value  
  lbmpdm.h, 612  
lbmpdm\_msg\_set\_field\_value\_vec  
  lbmpdm.h, 612  
lbmpdm\_msg\_set\_incl\_defn\_flag  
  lbmpdm.h, 612  
lbmpdm\_msg\_unset\_incl\_defn\_flag  
  lbmpdm.h, 613  
lbmpdm\_timestamp\_t, 287  
  tv\_secs, 287  
  tv\_usecs, 287  
lbmsdm.h, 614  
  lbmsdm\_decimal\_t, 650  
  LBMSDM\_ERR\_ADDING\_-  
    FIELD, 653  
  LBMSDM\_ERR\_BAD\_TYPE, 652  
  LBMSDM\_ERR\_CANNOT\_-  
    CONVERT, 652  
  LBMSDM\_ERR\_DELETING\_-  
    FIELD, 653  
  LBMSDM\_ERR\_DUPLICATE\_-  
    FIELD, 652  
  LBMSDM\_ERR\_EINVAL, 652  
  LBMSDM\_ERR\_ELEMENT\_-  
    NOT\_FOUND, 653  
  LBMSDM\_ERR\_ENOMEM, 652  
  LBMSDM\_ERR\_FIELD\_IS\_-  
    NULL, 653  
  LBMSDM\_ERR\_FIELD\_NOT\_-  
    FOUND, 652  
  LBMSDM\_ERR\_INVALID\_-  
    FIELD\_NAME, 653  
  LBMSDM\_ERR\_ITERATOR\_-  
    INVALID, 653  
  LBMSDM\_ERR\_MSG\_INVALID,  
    652  
  LBMSDM\_ERR\_-  
    NAMETOOLONG, 652  
  LBMSDM\_ERR\_NOT\_ARRAY,  
    652  
  LBMSDM\_ERR\_NOT\_SCALAR,  
    653  
  LBMSDM\_ERR\_TYPE\_-  
    MISMATCH, 653  
  LBMSDM\_ERR\_TYPE\_NOT\_-  
    SUPPORTED, 653

**LBMSDM\_ERR\_UNICODE\_-CONVERSION**, [653](#)  
**lbmsdm\_errmsg**, [653](#)  
**lbmsdm\_errnum**, [653](#)  
**LBMSDM\_FAILURE**, [652](#)  
**LBMSDM\_FIELD\_IS\_NULL**, [652](#)  
**LBMSDM\_INSUFFICIENT\_-BUFFER\_LENGTH**, [652](#)  
**lbmsdm\_iter\_create**, [653](#)  
**lbmsdm\_iter\_del**, [654](#)  
**lbmsdm\_iter\_del\_elem**, [654](#)  
**lbmsdm\_iter\_destroy**, [654](#)  
**lbmsdm\_iter\_first**, [654](#)  
**lbmsdm\_iter\_get\_elemcnt**, [655](#)  
**lbmsdm\_iter\_get\_elemlen**, [655](#)  
**lbmsdm\_iter\_get\_len**, [655](#)  
**lbmsdm\_iter\_get\_name**, [656](#)  
**lbmsdm\_iter\_get\_type**, [656](#)  
**lbmsdm\_iter\_is\_null**, [656](#)  
**lbmsdm\_iter\_next**, [656](#)  
**lbmsdm\_iter\_set\_null**, [657](#)  
**lbmsdm\_msg\_attr\_create**, [657](#)  
**lbmsdm\_msg\_attr\_delete**, [657](#)  
**lbmsdm\_msg\_attr\_dup**, [657](#)  
**lbmsdm\_msg\_attr\_getopt**, [658](#)  
**lbmsdm\_msg\_attr\_setopt**, [658](#)  
**lbmsdm\_msg\_attr\_str\_getopt**, [659](#)  
**lbmsdm\_msg\_attr\_str\_setopt**, [659](#)  
**lbmsdm\_msg\_clear**, [659](#)  
**lbmsdm\_msg\_clone**, [660](#)  
**lbmsdm\_msg\_create**, [660](#)  
**lbmsdm\_msg\_create\_ex**, [660](#)  
**lbmsdm\_msg\_del\_elem\_idx**, [661](#)  
**lbmsdm\_msg\_del\_elem\_name**, [661](#)  
**lbmsdm\_msg\_del\_idx**, [661](#)  
**lbmsdm\_msg\_del\_name**, [661](#)  
**lbmsdm\_msg\_destroy**, [662](#)  
**lbmsdm\_msg\_dump**, [662](#)  
**lbmsdm\_msg\_get\_data**, [662](#)  
**lbmsdm\_msg\_get\_datalen**, [663](#)  
**lbmsdm\_msg\_get\_elemcnt\_idx**, [663](#)  
**lbmsdm\_msg\_get\_elemcnt\_name**, [663](#)  
**lbmsdm\_msg\_get\_elemlen\_idx**, [664](#)  
**lbmsdm\_msg\_get\_elemlen\_name**, [664](#)  
**lbmsdm\_msg\_get\_fldcnt**, [664](#)  
**lbmsdm\_msg\_get\_idx\_name**, [665](#)  
**lbmsdm\_msg\_get\_len\_idx**, [665](#)  
**lbmsdm\_msg\_get\_len\_name**, [665](#)  
**lbmsdm\_msg\_get\_name\_idx**, [666](#)  
**lbmsdm\_msg\_get\_type\_idx**, [666](#)  
**lbmsdm\_msg\_get\_type\_name**, [666](#)  
**lbmsdm\_msg\_is\_null\_idx**, [667](#)  
**lbmsdm\_msg\_is\_null\_name**, [667](#)  
**lbmsdm\_msg\_parse**, [667](#)  
**lbmsdm\_msg\_parse\_ex**, [668](#)  
**lbmsdm\_msg\_parse\_reuse**, [668](#)  
**lbmsdm\_msg\_set\_null\_idx**, [668](#)  
**lbmsdm\_msg\_set\_null\_name**, [669](#)  
**LBMSDM\_NO\_MORE\_FIELDS**, [652](#)  
**LBMSDM\_SUCCESS**, [652](#)  
**LBMSDM\_TYPE\_ARRAY\_BLOB**, [652](#)  
**LBMSDM\_TYPE\_ARRAY\_-BOOLEAN**, [651](#)  
**LBMSDM\_TYPE\_ARRAY\_-DECIMAL**, [651](#)  
**LBMSDM\_TYPE\_ARRAY\_-DOUBLE**, [651](#)  
**LBMSDM\_TYPE\_ARRAY\_-FLOAT**, [651](#)  
**LBMSDM\_TYPE\_ARRAY\_INT16**, [651](#)  
**LBMSDM\_TYPE\_ARRAY\_INT32**, [651](#)  
**LBMSDM\_TYPE\_ARRAY\_INT64**, [651](#)  
**LBMSDM\_TYPE\_ARRAY\_INT8**, [651](#)  
**LBMSDM\_TYPE\_ARRAY\_-MESSAGE**, [652](#)  
**LBMSDM\_TYPE\_ARRAY\_-STRING**, [652](#)  
**LBMSDM\_TYPE\_ARRAY\_-TIMESTAMP**, [651](#)  
**LBMSDM\_TYPE\_ARRAY\_-UINT16**, [651](#)  
**LBMSDM\_TYPE\_ARRAY\_-UINT32**, [651](#)

LBMSDM\_TYPE\_ARRAY\_-  
  UINT64, 651  
LBMSDM\_TYPE\_ARRAY\_-  
  UINT8, 651  
LBMSDM\_TYPE\_ARRAY\_-  
  UNICODE, 652  
LBMSDM\_TYPE\_BLOB, 651  
LBMSDM\_TYPE\_BOOLEAN, 650  
LBMSDM\_TYPE\_DECIMAL, 651  
LBMSDM\_TYPE\_DOUBLE, 651  
LBMSDM\_TYPE\_FLOAT, 651  
LBMSDM\_TYPE\_INT16, 650  
LBMSDM\_TYPE\_INT32, 651  
LBMSDM\_TYPE\_INT64, 651  
LBMSDM\_TYPE\_INT8, 650  
LBMSDM\_TYPE\_INVALID, 650  
LBMSDM\_TYPE\_MESSAGE, 651  
LBMSDM\_TYPE\_STRING, 651  
LBMSDM\_TYPE\_TIMESTAMP,  
  651  
LBMSDM\_TYPE\_UINT16, 650  
LBMSDM\_TYPE\_UINT32, 651  
LBMSDM\_TYPE\_UINT64, 651  
LBMSDM\_TYPE\_UINT8, 650  
LBMSDM\_TYPE\_UNICODE, 651  
  lbmsdm\_win32\_static\_init, 669  
lbmsdm\_decimal\_t  
  lbmsdm.h, 650  
lbmsdm\_decimal\_t\_stct, 288  
  exp, 288  
  mant, 288  
LBMSDM\_ERR\_ADDING\_FIELD  
  lbmsdm.h, 653  
LBMSDM\_ERR\_BAD\_TYPE  
  lbmsdm.h, 652  
LBMSDM\_ERR\_CANNOT\_CONVERT  
  lbmsdm.h, 652  
LBMSDM\_ERR\_DELETING\_FIELD  
  lbmsdm.h, 653  
LBMSDM\_ERR\_DUPLICATE\_FIELD  
  lbmsdm.h, 652  
LBMSDM\_ERR\_EINVAL  
  lbmsdm.h, 652  
LBMSDM\_ERR\_ELEMENT\_NOT\_-  
  FOUND  
  lbmsdm.h, 653  
LBMSDM\_ERR\_ENOMEM  
  lbmsdm.h, 652  
LBMSDM\_ERR\_FIELD\_IS\_NULL  
  lbmsdm.h, 653  
LBMSDM\_ERR\_FIELD\_NOT\_FOUND  
  lbmsdm.h, 652  
LBMSDM\_ERR\_INVALID\_FIELD\_-  
  NAME  
  lbmsdm.h, 653  
LBMSDM\_ERR\_ITERATOR\_INVALID  
  lbmsdm.h, 653  
LBMSDM\_ERR\_MSG\_INVALID  
  lbmsdm.h, 652  
LBMSDM\_ERR\_NAMETOOLONG  
  lbmsdm.h, 652  
LBMSDM\_ERR\_NOT\_ARRAY  
  lbmsdm.h, 652  
LBMSDM\_ERR\_NOT\_SCALAR  
  lbmsdm.h, 653  
LBMSDM\_ERR\_TYPE\_MISMATCH  
  lbmsdm.h, 653  
LBMSDM\_ERR\_TYPE\_NOT\_-  
  SUPPORTED  
  lbmsdm.h, 653  
LBMSDM\_ERR\_UNICODE\_-  
  CONVERSION  
  lbmsdm.h, 653  
lbmsdm\_errmsg  
  lbmsdm.h, 653  
lbmsdm\_errnum  
  lbmsdm.h, 653  
LBMSDM\_FAILURE  
  lbmsdm.h, 652  
LBMSDM\_FIELD\_IS\_NULL  
  lbmsdm.h, 652  
LBMSDM\_INSUFFICIENT\_BUFFER\_-  
  LENGTH  
  lbmsdm.h, 652  
lbmsdm\_iter\_add\_blob\_elem  
  add\_elem\_iter, 35  
lbmsdm\_iter\_add\_boolean\_elem  
  add\_elem\_iter, 35  
lbmsdm\_iter\_add\_decimal\_elem  
  add\_elem\_iter, 35  
lbmsdm\_iter\_add\_double\_elem  
  add\_elem\_iter, 35

lbmsdm\_iter\_add\_float\_elem  
     add\_elem\_iter, 36  
 lbmsdm\_iter\_add\_int16\_elem  
     add\_elem\_iter, 36  
 lbmsdm\_iter\_add\_int32\_elem  
     add\_elem\_iter, 36  
 lbmsdm\_iter\_add\_int64\_elem  
     add\_elem\_iter, 36  
 lbmsdm\_iter\_add\_int8\_elem  
     add\_elem\_iter, 36  
 lbmsdm\_iter\_add\_message\_elem  
     add\_elem\_iter, 36  
 lbmsdm\_iter\_add\_string\_elem  
     add\_elem\_iter, 36  
 lbmsdm\_iter\_add\_timestamp\_elem  
     add\_elem\_iter, 37  
 lbmsdm\_iter\_add\_uint16\_elem  
     add\_elem\_iter, 37  
 lbmsdm\_iter\_add\_uint32\_elem  
     add\_elem\_iter, 37  
 lbmsdm\_iter\_add\_uint64\_elem  
     add\_elem\_iter, 37  
 lbmsdm\_iter\_add\_uint8\_elem  
     add\_elem\_iter, 37  
 lbmsdm\_iter\_add\_unicode\_elem  
     add\_elem\_iter, 37  
 lbmsdm\_iter\_create  
     lbmsdm.h, 653  
 lbmsdm\_iter\_del  
     lbmsdm.h, 654  
 lbmsdm\_iter\_del\_elem  
     lbmsdm.h, 654  
 lbmsdm\_iter\_destroy  
     lbmsdm.h, 654  
 lbmsdm\_iter\_first  
     lbmsdm.h, 654  
 lbmsdm\_iter\_get\_blob  
     get\_scalar\_iter, 50  
 lbmsdm\_iter\_get\_blob\_elem  
     get\_elem\_iter, 66  
 lbmsdm\_iter\_get\_boolean  
     get\_scalar\_iter, 50  
 lbmsdm\_iter\_get\_boolean\_elem  
     get\_elem\_iter, 66  
 lbmsdm\_iter\_get\_decimal  
     get\_scalar\_iter, 50  
 lbmsdm\_iter\_get\_decimal\_elem  
     get\_elem\_iter, 66  
 lbmsdm\_iter\_get\_double  
     get\_scalar\_iter, 50  
 lbmsdm\_iter\_get\_double\_elem  
     get\_elem\_iter, 67  
 lbmsdm\_iter\_get\_elementnt  
     lbmsdm.h, 655  
 lbmsdm\_iter\_get\_elemlen  
     lbmsdm.h, 655  
 lbmsdm\_iter\_get\_float  
     get\_scalar\_iter, 51  
 lbmsdm\_iter\_get\_float\_elem  
     get\_elem\_iter, 67  
 lbmsdm\_iter\_get\_int16  
     get\_scalar\_iter, 51  
 lbmsdm\_iter\_get\_int16\_elem  
     get\_elem\_iter, 67  
 lbmsdm\_iter\_get\_int32  
     get\_scalar\_iter, 51  
 lbmsdm\_iter\_get\_int32\_elem  
     get\_elem\_iter, 67  
 lbmsdm\_iter\_get\_int64  
     get\_scalar\_iter, 51  
 lbmsdm\_iter\_get\_int64\_elem  
     get\_elem\_iter, 67  
 lbmsdm\_iter\_get\_int8  
     get\_scalar\_iter, 51  
 lbmsdm\_iter\_get\_int8\_elem  
     get\_elem\_iter, 67  
 lbmsdm\_iter\_get\_len  
     lbmsdm.h, 655  
 lbmsdm\_iter\_get\_message  
     get\_scalar\_iter, 51  
 lbmsdm\_iter\_get\_message\_elem  
     get\_elem\_iter, 67  
 lbmsdm\_iter\_get\_name  
     lbmsdm.h, 656  
 lbmsdm\_iter\_get\_string  
     get\_scalar\_iter, 51  
 lbmsdm\_iter\_get\_string\_elem  
     get\_elem\_iter, 68  
 lbmsdm\_iter\_get\_timestamp  
     get\_scalar\_iter, 52  
 lbmsdm\_iter\_get\_timestamp\_elem  
     get\_elem\_iter, 68

lbmsdm\_iter\_get\_type  
    lbmsdm.h, 656

lbmsdm\_iter\_get\_uint16  
    get\_scalar\_iter, 52

lbmsdm\_iter\_get\_uint16\_elem  
    get\_elem\_iter, 68

lbmsdm\_iter\_get\_uint32  
    get\_scalar\_iter, 52

lbmsdm\_iter\_get\_uint32\_elem  
    get\_elem\_iter, 68

lbmsdm\_iter\_get\_uint64  
    get\_scalar\_iter, 52

lbmsdm\_iter\_get\_uint64\_elem  
    get\_elem\_iter, 68

lbmsdm\_iter\_get\_uint8  
    get\_scalar\_iter, 52

lbmsdm\_iter\_get\_uint8\_elem  
    get\_elem\_iter, 69

lbmsdm\_iter\_get\_unicode  
    get\_scalar\_iter, 53

lbmsdm\_iter\_get\_unicode\_elem  
    get\_elem\_iter, 69

lbmsdm\_iter\_is\_null  
    lbmsdm.h, 656

lbmsdm\_iter\_next  
    lbmsdm.h, 656

lbmsdm\_iter\_set\_blob  
    set\_iter, 81

lbmsdm\_iter\_set\_blob\_array  
    set\_array\_iter, 93

lbmsdm\_iter\_set\_blob\_elem  
    set\_elem\_iter, 108

lbmsdm\_iter\_set\_boolean  
    set\_iter, 81

lbmsdm\_iter\_set\_boolean\_array  
    set\_array\_iter, 93

lbmsdm\_iter\_set\_boolean\_elem  
    set\_elem\_iter, 108

lbmsdm\_iter\_set\_decimal  
    set\_iter, 81

lbmsdm\_iter\_set\_decimal\_array  
    set\_array\_iter, 94

lbmsdm\_iter\_set\_decimal\_elem  
    set\_elem\_iter, 108

lbmsdm\_iter\_set\_double  
    set\_iter, 81

lbmsdm\_iter\_set\_double\_array  
    set\_array\_iter, 94

lbmsdm\_iter\_set\_double\_elem  
    set\_elem\_iter, 108

lbmsdm\_iter\_set\_float  
    set\_iter, 82

lbmsdm\_iter\_set\_float\_array  
    set\_array\_iter, 94

lbmsdm\_iter\_set\_float\_elem  
    set\_elem\_iter, 109

lbmsdm\_iter\_set\_int16  
    set\_iter, 82

lbmsdm\_iter\_set\_int16\_array  
    set\_array\_iter, 94

lbmsdm\_iter\_set\_int16\_elem  
    set\_elem\_iter, 109

lbmsdm\_iter\_set\_int32  
    set\_iter, 82

lbmsdm\_iter\_set\_int32\_array  
    set\_array\_iter, 94

lbmsdm\_iter\_set\_int32\_elem  
    set\_elem\_iter, 109

lbmsdm\_iter\_set\_int64  
    set\_iter, 82

lbmsdm\_iter\_set\_int64\_array  
    set\_array\_iter, 94

lbmsdm\_iter\_set\_int64\_elem  
    set\_elem\_iter, 109

lbmsdm\_iter\_set\_int8  
    set\_iter, 82

lbmsdm\_iter\_set\_int8\_array  
    set\_array\_iter, 95

lbmsdm\_iter\_set\_int8\_elem  
    set\_elem\_iter, 109

lbmsdm\_iter\_set\_message  
    set\_iter, 82

lbmsdm\_iter\_set\_message\_array  
    set\_array\_iter, 95

lbmsdm\_iter\_set\_message\_elem  
    set\_elem\_iter, 109

lbmsdm\_iter\_set\_null  
    lbmsdm.h, 657

lbmsdm\_iter\_set\_string  
    set\_iter, 82

lbmsdm\_iter\_set\_string\_array  
    set\_array\_iter, 95

lbmsdm\_iter\_set\_string\_elem  
     set\_elem\_iter, 109  
 lbmsdm\_iter\_set\_timestamp  
     set\_iter, 83  
 lbmsdm\_iter\_set\_timestamp\_array  
     set\_array\_iter, 95  
 lbmsdm\_iter\_set\_timestamp\_elem  
     set\_elem\_iter, 110  
 lbmsdm\_iter\_set\_uint16  
     set\_iter, 83  
 lbmsdm\_iter\_set\_uint16\_array  
     set\_array\_iter, 95  
 lbmsdm\_iter\_set\_uint16\_elem  
     set\_elem\_iter, 110  
 lbmsdm\_iter\_set\_uint32  
     set\_iter, 83  
 lbmsdm\_iter\_set\_uint32\_array  
     set\_array\_iter, 95  
 lbmsdm\_iter\_set\_uint32\_elem  
     set\_elem\_iter, 110  
 lbmsdm\_iter\_set\_uint64  
     set\_iter, 83  
 lbmsdm\_iter\_set\_uint64\_array  
     set\_array\_iter, 95  
 lbmsdm\_iter\_set\_uint64\_elem  
     set\_elem\_iter, 110  
 lbmsdm\_iter\_set\_uint8  
     set\_iter, 83  
 lbmsdm\_iter\_set\_uint8\_array  
     set\_array\_iter, 96  
 lbmsdm\_iter\_set\_uint8\_elem  
     set\_elem\_iter, 110  
 lbmsdm\_iter\_set\_unicode  
     set\_iter, 83  
 lbmsdm\_iter\_set\_unicode\_array  
     set\_array\_iter, 96  
 lbmsdm\_iter\_set\_unicode\_elem  
     set\_elem\_iter, 110  
 lbmsdm\_msg\_add\_blob  
     add, 16  
 lbmsdm\_msg\_add\_blob\_array  
     add\_array, 21  
 lbmsdm\_msg\_add\_blob\_elem\_idx  
     add\_elem\_idx, 25  
 lbmsdm\_msg\_add\_blob\_elem\_name  
     add\_elem\_name, 30  
 lbmsdm\_msg\_add\_boolean  
     add, 16  
 lbmsdm\_msg\_add\_boolean\_array  
     add\_array, 21  
 lbmsdm\_msg\_add\_boolean\_elem\_idx  
     add\_elem\_idx, 25  
 lbmsdm\_msg\_add\_boolean\_elem\_name  
     add\_elem\_name, 30  
 lbmsdm\_msg\_add\_decimal  
     add, 17  
 lbmsdm\_msg\_add\_decimal\_array  
     add\_array, 21  
 lbmsdm\_msg\_add\_decimal\_elem\_idx  
     add\_elem\_idx, 25  
 lbmsdm\_msg\_add\_decimal\_elem\_name  
     add\_elem\_name, 30  
 lbmsdm\_msg\_add\_double  
     add, 17  
 lbmsdm\_msg\_add\_double\_array  
     add\_array, 21  
 lbmsdm\_msg\_add\_double\_elem\_idx  
     add\_elem\_idx, 25  
 lbmsdm\_msg\_add\_double\_elem\_name  
     add\_elem\_name, 30  
 lbmsdm\_msg\_add\_float  
     add, 17  
 lbmsdm\_msg\_add\_float\_array  
     add\_array, 21  
 lbmsdm\_msg\_add\_float\_elem\_idx  
     add\_elem\_idx, 26  
 lbmsdm\_msg\_add\_float\_elem\_name  
     add\_elem\_name, 31  
 lbmsdm\_msg\_add\_int16  
     add, 17  
 lbmsdm\_msg\_add\_int16\_array  
     add\_array, 21  
 lbmsdm\_msg\_add\_int16\_elem\_idx  
     add\_elem\_idx, 26  
 lbmsdm\_msg\_add\_int16\_elem\_name  
     add\_elem\_name, 31  
 lbmsdm\_msg\_add\_int32  
     add, 17  
 lbmsdm\_msg\_add\_int32\_array  
     add\_array, 21  
 lbmsdm\_msg\_add\_int32\_elem\_idx  
     add\_elem\_idx, 26

lbmsdm\_msg\_add\_int32\_elem\_name  
    add\_elem\_name, 31

lbmsdm\_msg\_add\_int64  
    add, 17

lbmsdm\_msg\_add\_int64\_array  
    add\_array, 22

lbmsdm\_msg\_add\_int64\_elem\_idx  
    add\_elem\_idx, 26

lbmsdm\_msg\_add\_int64\_elem\_name  
    add\_elem\_name, 31

lbmsdm\_msg\_add\_int8  
    add, 18

lbmsdm\_msg\_add\_int8\_array  
    add\_array, 22

lbmsdm\_msg\_add\_int8\_elem\_idx  
    add\_elem\_idx, 26

lbmsdm\_msg\_add\_int8\_elem\_name  
    add\_elem\_name, 31

lbmsdm\_msg\_add\_message  
    add, 18

lbmsdm\_msg\_add\_message\_array  
    add\_array, 22

lbmsdm\_msg\_add\_message\_elem\_idx  
    add\_elem\_idx, 26

lbmsdm\_msg\_add\_message\_elem\_name  
    add\_elem\_name, 31

lbmsdm\_msg\_add\_string  
    add, 18

lbmsdm\_msg\_add\_string\_array  
    add\_array, 22

lbmsdm\_msg\_add\_string\_elem\_idx  
    add\_elem\_idx, 26

lbmsdm\_msg\_add\_string\_elem\_name  
    add\_elem\_name, 31

lbmsdm\_msg\_add\_timestamp  
    add, 18

lbmsdm\_msg\_add\_timestamp\_array  
    add\_array, 22

lbmsdm\_msg\_add\_timestamp\_elem\_idx  
    add\_elem\_idx, 27

lbmsdm\_msg\_add\_timestamp\_elem\_-  
    name  
        add\_elem\_name, 32

lbmsdm\_msg\_add\_uint16  
    add, 18

lbmsdm\_msg\_add\_uint16\_array  
    add\_array, 22

lbmsdm\_msg\_add\_uint16\_elem\_idx  
    add\_elem\_idx, 27

lbmsdm\_msg\_add\_uint16\_elem\_name  
    add\_elem\_name, 32

lbmsdm\_msg\_add\_uint32  
    add, 18

lbmsdm\_msg\_add\_uint32\_array  
    add\_array, 22

lbmsdm\_msg\_add\_uint32\_elem\_idx  
    add\_elem\_idx, 27

lbmsdm\_msg\_add\_uint32\_elem\_name  
    add\_elem\_name, 32

lbmsdm\_msg\_add\_uint64  
    add, 18

lbmsdm\_msg\_add\_uint64\_array  
    add\_array, 23

lbmsdm\_msg\_add\_uint64\_elem\_idx  
    add\_elem\_idx, 27

lbmsdm\_msg\_add\_uint64\_elem\_name  
    add\_elem\_name, 32

lbmsdm\_msg\_add\_uint8  
    add, 19

lbmsdm\_msg\_add\_uint8\_array  
    add\_array, 23

lbmsdm\_msg\_add\_uint8\_elem\_idx  
    add\_elem\_idx, 27

lbmsdm\_msg\_add\_uint8\_elem\_name  
    add\_elem\_name, 32

lbmsdm\_msg\_add\_unicode  
    add, 19

lbmsdm\_msg\_add\_unicode\_array  
    add\_array, 23

lbmsdm\_msg\_add\_unicode\_elem\_idx  
    add\_elem\_idx, 27

lbmsdm\_msg\_add\_unicode\_elem\_name  
    add\_elem\_name, 32

lbmsdm\_msg\_attr\_create  
    lbmsdm.h, 657

lbmsdm\_msg\_attr\_delete  
    lbmsdm.h, 657

lbmsdm\_msg\_attr\_dup  
    lbmsdm.h, 657

lbmsdm\_msg\_attr\_getopt  
    lbmsdm.h, 658

lbmsdm\_msg\_attr\_setopt

lbmsdm.h, 658  
 lbmsdm\_msg\_attr\_str\_getopt  
     lbmsdm.h, 659  
 lbmsdm\_msg\_attr\_str\_setopt  
     lbmsdm.h, 659  
 lbmsdm\_msg\_clear  
     lbmsdm.h, 659  
 lbmsdm\_msg\_clone  
     lbmsdm.h, 660  
 lbmsdm\_msg\_create  
     lbmsdm.h, 660  
 lbmsdm\_msg\_create\_ex  
     lbmsdm.h, 660  
 lbmsdm\_msg\_del\_elem\_idx  
     lbmsdm.h, 661  
 lbmsdm\_msg\_del\_elem\_name  
     lbmsdm.h, 661  
 lbmsdm\_msg\_del\_idx  
     lbmsdm.h, 661  
 lbmsdm\_msg\_del\_name  
     lbmsdm.h, 661  
 lbmsdm\_msg\_destroy  
     lbmsdm.h, 662  
 lbmsdm\_msg\_dump  
     lbmsdm.h, 662  
 lbmsdm\_msg\_get\_blob\_elem\_idx  
     get\_elem\_idx, 55  
 lbmsdm\_msg\_get\_blob\_elem\_name  
     get\_elem\_name, 60  
 lbmsdm\_msg\_get\_blob\_idx  
     get\_scalar\_idx, 40  
 lbmsdm\_msg\_get\_blob\_name  
     get\_scalar\_name, 45  
 lbmsdm\_msg\_get\_boolean\_elem\_idx  
     get\_elem\_idx, 55  
 lbmsdm\_msg\_get\_boolean\_elem\_name  
     get\_elem\_name, 60  
 lbmsdm\_msg\_get\_boolean\_idx  
     get\_scalar\_idx, 40  
 lbmsdm\_msg\_get\_boolean\_name  
     get\_scalar\_name, 45  
 lbmsdm\_msg\_get\_data  
     lbmsdm.h, 662  
 lbmsdm\_msg\_get\_datalen  
     lbmsdm.h, 663  
 lbmsdm\_msg\_get\_decimal\_elem\_idx  
     get\_elem\_idx, 55  
 lbmsdm\_msg\_get\_decimal\_elem\_name  
     get\_elem\_name, 60  
 lbmsdm\_msg\_get\_decimal\_idx  
     get\_scalar\_idx, 40  
 lbmsdm\_msg\_get\_decimal\_name  
     get\_scalar\_name, 45  
 lbmsdm\_msg\_get\_double\_elem\_idx  
     get\_elem\_idx, 56  
 lbmsdm\_msg\_get\_double\_elem\_name  
     get\_elem\_name, 61  
 lbmsdm\_msg\_get\_double\_idx  
     get\_scalar\_idx, 41  
 lbmsdm\_msg\_get\_double\_name  
     get\_scalar\_name, 46  
 lbmsdm\_msg\_get\_element\_idx  
     lbmsdm.h, 663  
 lbmsdm\_msg\_get\_element\_name  
     lbmsdm.h, 663  
 lbmsdm\_msg\_get\_elemlen\_idx  
     lbmsdm.h, 664  
 lbmsdm\_msg\_get\_elemlen\_name  
     lbmsdm.h, 664  
 lbmsdm\_msg\_get\_fldcnt  
     lbmsdm.h, 664  
 lbmsdm\_msg\_get\_float\_elem\_idx  
     get\_elem\_idx, 56  
 lbmsdm\_msg\_get\_float\_elem\_name  
     get\_elem\_name, 61  
 lbmsdm\_msg\_get\_float\_idx  
     get\_scalar\_idx, 41  
 lbmsdm\_msg\_get\_float\_name  
     get\_scalar\_name, 46  
 lbmsdm\_msg\_get\_idx\_name  
     lbmsdm.h, 665  
 lbmsdm\_msg\_get\_int16\_elem\_idx  
     get\_elem\_idx, 56  
 lbmsdm\_msg\_get\_int16\_elem\_name  
     get\_elem\_name, 61  
 lbmsdm\_msg\_get\_int16\_idx  
     get\_scalar\_idx, 41  
 lbmsdm\_msg\_get\_int16\_name  
     get\_scalar\_name, 46  
 lbmsdm\_msg\_get\_int32\_elem\_idx  
     get\_elem\_idx, 56  
 lbmsdm\_msg\_get\_int32\_elem\_name

get\_elem\_name, 61  
lbmsdm\_msg\_get\_int32\_idx  
    get\_scalar\_idx, 41  
lbmsdm\_msg\_get\_int32\_name  
    get\_scalar\_name, 46  
lbmsdm\_msg\_get\_int64\_elem\_idx  
    get\_elem\_idx, 56  
lbmsdm\_msg\_get\_int64\_elem\_name  
    get\_elem\_name, 61  
lbmsdm\_msg\_get\_int64\_idx  
    get\_scalar\_idx, 41  
lbmsdm\_msg\_get\_int64\_name  
    get\_scalar\_name, 46  
lbmsdm\_msg\_get\_int8\_elem\_idx  
    get\_elem\_idx, 56  
lbmsdm\_msg\_get\_int8\_elem\_name  
    get\_elem\_name, 61  
lbmsdm\_msg\_get\_int8\_idx  
    get\_scalar\_idx, 41  
lbmsdm\_msg\_get\_int8\_name  
    get\_scalar\_name, 46  
lbmsdm\_msg\_get\_len\_idx  
    lbmsdm.h, 665  
lbmsdm\_msg\_get\_len\_name  
    lbmsdm.h, 665  
lbmsdm\_msg\_get\_message\_elem\_idx  
    get\_elem\_idx, 56  
lbmsdm\_msg\_get\_message\_elem\_name  
    get\_elem\_name, 62  
lbmsdm\_msg\_get\_message\_idx  
    get\_scalar\_idx, 41  
lbmsdm\_msg\_get\_message\_name  
    get\_scalar\_name, 46  
lbmsdm\_msg\_get\_name\_idx  
    lbmsdm.h, 666  
lbmsdm\_msg\_get\_string\_elem\_idx  
    get\_elem\_idx, 57  
lbmsdm\_msg\_get\_string\_elem\_name  
    get\_elem\_name, 62  
lbmsdm\_msg\_get\_string\_idx  
    get\_scalar\_idx, 42  
lbmsdm\_msg\_get\_string\_name  
    get\_scalar\_name, 47  
lbmsdm\_msg\_get\_timestamp\_elem\_idx  
    get\_elem\_idx, 57  
lbmsdm\_msg\_get\_timestamp\_elem\_name  
    get\_elem\_name, 62  
lbmsdm\_msg\_get\_timestamp\_idx  
    get\_scalar\_idx, 42  
lbmsdm\_msg\_get\_timestamp\_name  
    get\_scalar\_name, 47  
lbmsdm\_msg\_get\_type\_idx  
    lbmsdm.h, 666  
lbmsdm\_msg\_get\_type\_name  
    lbmsdm.h, 666  
lbmsdm\_msg\_get\_uint16\_elem\_idx  
    get\_elem\_idx, 57  
lbmsdm\_msg\_get\_uint16\_elem\_name  
    get\_elem\_name, 62  
lbmsdm\_msg\_get\_uint16\_idx  
    get\_scalar\_idx, 42  
lbmsdm\_msg\_get\_uint16\_name  
    get\_scalar\_name, 47  
lbmsdm\_msg\_get\_uint32\_elem\_idx  
    get\_elem\_idx, 57  
lbmsdm\_msg\_get\_uint32\_elem\_name  
    get\_elem\_name, 63  
lbmsdm\_msg\_get\_uint32\_idx  
    get\_scalar\_idx, 42  
lbmsdm\_msg\_get\_uint32\_name  
    get\_scalar\_name, 47  
lbmsdm\_msg\_get\_uint64\_elem\_idx  
    get\_elem\_idx, 58  
lbmsdm\_msg\_get\_uint64\_elem\_name  
    get\_elem\_name, 63  
lbmsdm\_msg\_get\_uint64\_idx  
    get\_scalar\_idx, 42  
lbmsdm\_msg\_get\_uint64\_name  
    get\_scalar\_name, 47  
lbmsdm\_msg\_get\_uint8\_elem\_idx  
    get\_elem\_idx, 58  
lbmsdm\_msg\_get\_uint8\_elem\_name  
    get\_elem\_name, 63  
lbmsdm\_msg\_get\_uint8\_idx  
    get\_scalar\_idx, 43  
lbmsdm\_msg\_get\_uint8\_name  
    get\_scalar\_name, 48  
lbmsdm\_msg\_get\_unicode\_elem\_idx  
    get\_elem\_idx, 58  
lbmsdm\_msg\_get\_unicode\_elem\_name

lbmsdm\_msg\_get\_unicode\_idx  
     get\_scalar\_idx, 43  
 lbmsdm\_msg\_get\_unicode\_name  
     get\_scalar\_name, 48  
 lbmsdm\_msg\_is\_null\_idx  
     lbmsdm.h, 667  
 lbmsdm\_msg\_is\_null\_name  
     lbmsdm.h, 667  
 lbmsdm\_msg\_parse  
     lbmsdm.h, 667  
 lbmsdm\_msg\_parse\_ex  
     lbmsdm.h, 668  
 lbmsdm\_msg\_parse\_reuse  
     lbmsdm.h, 668  
 lbmsdm\_msg\_set\_blob\_array\_idx  
     set\_array\_idx, 86  
 lbmsdm\_msg\_set\_blob\_array\_name  
     set\_array\_name, 90  
 lbmsdm\_msg\_set\_blob\_elem\_idx  
     set\_elem\_idx, 98  
 lbmsdm\_msg\_set\_blob\_elem\_name  
     set\_elem\_name, 103  
 lbmsdm\_msg\_set\_blob\_idx  
     set\_idx, 71  
 lbmsdm\_msg\_set\_blob\_name  
     set\_name, 76  
 lbmsdm\_msg\_set\_boolean\_array\_idx  
     set\_array\_idx, 86  
 lbmsdm\_msg\_set\_boolean\_array\_name  
     set\_array\_name, 90  
 lbmsdm\_msg\_set\_boolean\_elem\_idx  
     set\_elem\_idx, 98  
 lbmsdm\_msg\_set\_boolean\_elem\_name  
     set\_elem\_name, 103  
 lbmsdm\_msg\_set\_boolean\_idx  
     set\_idx, 71  
 lbmsdm\_msg\_set\_boolean\_name  
     set\_name, 76  
 lbmsdm\_msg\_set\_decimal\_array\_idx  
     set\_array\_idx, 86  
 lbmsdm\_msg\_set\_decimal\_array\_name  
     set\_array\_name, 90  
 lbmsdm\_msg\_set\_decimal\_elem\_idx  
     set\_elem\_idx, 98  
 lbmsdm\_msg\_set\_decimal\_elem\_name  
     set\_elem\_name, 103  
 lbmsdm\_msg\_set\_decimal\_idx  
     set\_idx, 71  
 lbmsdm\_msg\_set\_decimal\_name  
     set\_name, 76  
 lbmsdm\_msg\_set\_double\_array\_idx  
     set\_array\_idx, 86  
 lbmsdm\_msg\_set\_double\_array\_name  
     set\_array\_name, 90  
 lbmsdm\_msg\_set\_double\_elem\_idx  
     set\_elem\_idx, 98  
 lbmsdm\_msg\_set\_double\_elem\_name  
     set\_elem\_name, 104  
 lbmsdm\_msg\_set\_double\_idx  
     set\_idx, 71  
 lbmsdm\_msg\_set\_double\_name  
     set\_name, 76  
 lbmsdm\_msg\_set\_float\_array\_idx  
     set\_array\_idx, 86  
 lbmsdm\_msg\_set\_float\_array\_name  
     set\_array\_name, 90  
 lbmsdm\_msg\_set\_float\_elem\_idx  
     set\_elem\_idx, 99  
 lbmsdm\_msg\_set\_float\_elem\_name  
     set\_elem\_name, 104  
 lbmsdm\_msg\_set\_float\_idx  
     set\_idx, 72  
 lbmsdm\_msg\_set\_float\_name  
     set\_name, 77  
 lbmsdm\_msg\_set\_int16\_array\_idx  
     set\_array\_idx, 86  
 lbmsdm\_msg\_set\_int16\_array\_name  
     set\_array\_name, 90  
 lbmsdm\_msg\_set\_int16\_elem\_idx  
     set\_elem\_idx, 99  
 lbmsdm\_msg\_set\_int16\_elem\_name  
     set\_elem\_name, 104  
 lbmsdm\_msg\_set\_int16\_idx  
     set\_idx, 72  
 lbmsdm\_msg\_set\_int16\_name  
     set\_name, 77  
 lbmsdm\_msg\_set\_int32\_array\_idx  
     set\_array\_idx, 87  
 lbmsdm\_msg\_set\_int32\_array\_name  
     set\_array\_name, 91  
 lbmsdm\_msg\_set\_int32\_elem\_idx

set\_elem\_idx, 99  
lbmsdm\_msg\_set\_int32\_elem\_name  
    set\_elem\_name, 104  
lbmsdm\_msg\_set\_int32\_idx  
    set\_idx, 72  
lbmsdm\_msg\_set\_int32\_name  
    set\_name, 77  
lbmsdm\_msg\_set\_int64\_array\_idx  
    set\_array\_idx, 87  
lbmsdm\_msg\_set\_int64\_array\_name  
    set\_array\_name, 91  
lbmsdm\_msg\_set\_int64\_elem\_idx  
    set\_elem\_idx, 99  
lbmsdm\_msg\_set\_int64\_elem\_name  
    set\_elem\_name, 104  
lbmsdm\_msg\_set\_int64\_idx  
    set\_idx, 72  
lbmsdm\_msg\_set\_int64\_name  
    set\_name, 77  
lbmsdm\_msg\_set\_int8\_array\_idx  
    set\_array\_idx, 87  
lbmsdm\_msg\_set\_int8\_array\_name  
    set\_array\_name, 91  
lbmsdm\_msg\_set\_int8\_elem\_idx  
    set\_elem\_idx, 99  
lbmsdm\_msg\_set\_int8\_elem\_name  
    set\_elem\_name, 104  
lbmsdm\_msg\_set\_int8\_idx  
    set\_idx, 72  
lbmsdm\_msg\_set\_int8\_name  
    set\_name, 77  
lbmsdm\_msg\_set\_message\_array\_idx  
    set\_array\_idx, 87  
lbmsdm\_msg\_set\_message\_array\_name  
    set\_array\_name, 91  
lbmsdm\_msg\_set\_message\_elem\_idx  
    set\_elem\_idx, 99  
lbmsdm\_msg\_set\_message\_elem\_name  
    set\_elem\_name, 105  
lbmsdm\_msg\_set\_message\_idx  
    set\_idx, 72  
lbmsdm\_msg\_set\_message\_name  
    set\_name, 77  
lbmsdm\_msg\_set\_null\_idx  
    lbmsdm.h, 668  
lbmsdm\_msg\_set\_null\_name

lbmsdm.h, 669  
lbmsdm\_msg\_set\_string\_array\_idx  
    set\_array\_idx, 87  
lbmsdm\_msg\_set\_string\_array\_name  
    set\_array\_name, 91  
lbmsdm\_msg\_set\_string\_elem\_idx  
    set\_elem\_idx, 99  
lbmsdm\_msg\_set\_string\_elem\_name  
    set\_elem\_name, 105  
lbmsdm\_msg\_set\_string\_idx  
    set\_idx, 72  
lbmsdm\_msg\_set\_string\_name  
    set\_name, 77  
lbmsdm\_msg\_set\_timestamp\_array\_idx  
    set\_array\_idx, 87  
lbmsdm\_msg\_set\_timestamp\_array\_-  
    name  
    set\_array\_name, 91  
lbmsdm\_msg\_set\_timestamp\_elem\_idx  
    set\_elem\_idx, 100  
lbmsdm\_msg\_set\_timestamp\_elem\_-  
    name  
    set\_elem\_name, 105  
lbmsdm\_msg\_set\_timestamp\_idx  
    set\_idx, 73  
lbmsdm\_msg\_set\_timestamp\_name  
    set\_name, 78  
lbmsdm\_msg\_set\_uint16\_array\_idx  
    set\_array\_idx, 87  
lbmsdm\_msg\_set\_uint16\_array\_name  
    set\_array\_name, 91  
lbmsdm\_msg\_set\_uint16\_elem\_idx  
    set\_elem\_idx, 100  
lbmsdm\_msg\_set\_uint16\_elem\_name  
    set\_elem\_name, 105  
lbmsdm\_msg\_set\_uint16\_idx  
    set\_idx, 73  
lbmsdm\_msg\_set\_uint16\_name  
    set\_name, 78  
lbmsdm\_msg\_set\_uint32\_array\_idx  
    set\_array\_idx, 88  
lbmsdm\_msg\_set\_uint32\_array\_name  
    set\_array\_name, 92  
lbmsdm\_msg\_set\_uint32\_elem\_idx  
    set\_elem\_idx, 100  
lbmsdm\_msg\_set\_uint32\_elem\_name

set\_elem\_name, 105  
**lbmsdm\_msg\_set\_uint32\_idx**  
 set\_idx, 73  
**lbmsdm\_msg\_set\_uint32\_name**  
 set\_name, 78  
**lbmsdm\_msg\_set\_uint64\_array\_idx**  
 set\_array\_idx, 88  
**lbmsdm\_msg\_set\_uint64\_array\_name**  
 set\_array\_name, 92  
**lbmsdm\_msg\_set\_uint64\_elem\_idx**  
 set\_elem\_idx, 100  
**lbmsdm\_msg\_set\_uint64\_elem\_name**  
 set\_elem\_name, 105  
**lbmsdm\_msg\_set\_uint64\_idx**  
 set\_idx, 73  
**lbmsdm\_msg\_set\_uint64\_name**  
 set\_name, 78  
**lbmsdm\_msg\_set\_uint8\_array\_idx**  
 set\_array\_idx, 88  
**lbmsdm\_msg\_set\_uint8\_array\_name**  
 set\_array\_name, 92  
**lbmsdm\_msg\_set\_uint8\_elem\_idx**  
 set\_elem\_idx, 100  
**lbmsdm\_msg\_set\_uint8\_elem\_name**  
 set\_elem\_name, 106  
**lbmsdm\_msg\_set\_uint8\_idx**  
 set\_idx, 73  
**lbmsdm\_msg\_set\_uint8\_name**  
 set\_name, 78  
**lbmsdm\_msg\_set\_unicode\_array\_idx**  
 set\_array\_idx, 88  
**lbmsdm\_msg\_set\_unicode\_array\_name**  
 set\_array\_name, 92  
**lbmsdm\_msg\_set\_unicode\_elem\_idx**  
 set\_elem\_idx, 100  
**lbmsdm\_msg\_set\_unicode\_elem\_name**  
 set\_elem\_name, 106  
**lbmsdm\_msg\_set\_unicode\_idx**  
 set\_idx, 73  
**lbmsdm\_msg\_set\_unicode\_name**  
 set\_name, 78  
**LBMSDM\_NO\_MORE\_FIELDS**  
 lbmsdm.h, 652  
**LBMSDM\_SUCCESS**  
 lbmsdm.h, 652  
**LBMSDM\_TYPE\_ARRAY\_BLOB**  
 lbmsdm.h, 652  
**LBMSDM\_TYPE\_ARRAY\_BOOLEAN**  
 lbmsdm.h, 651  
**LBMSDM\_TYPE\_ARRAY\_DECIMAL**  
 lbmsdm.h, 651  
**LBMSDM\_TYPE\_ARRAY\_DOUBLE**  
 lbmsdm.h, 651  
**LBMSDM\_TYPE\_ARRAY\_FLOAT**  
 lbmsdm.h, 651  
**LBMSDM\_TYPE\_ARRAY\_INT16**  
 lbmsdm.h, 651  
**LBMSDM\_TYPE\_ARRAY\_INT32**  
 lbmsdm.h, 651  
**LBMSDM\_TYPE\_ARRAY\_INT64**  
 lbmsdm.h, 651  
**LBMSDM\_TYPE\_ARRAY\_INT8**  
 lbmsdm.h, 651  
**LBMSDM\_TYPE\_ARRAY\_MESSAGE**  
 lbmsdm.h, 652  
**LBMSDM\_TYPE\_ARRAY\_STRING**  
 lbmsdm.h, 652  
**LBMSDM\_TYPE\_ARRAY\_TIMESTAMP**  
 lbmsdm.h, 651  
**LBMSDM\_TYPE\_ARRAY\_UINT16**  
 lbmsdm.h, 651  
**LBMSDM\_TYPE\_ARRAY\_UINT32**  
 lbmsdm.h, 651  
**LBMSDM\_TYPE\_ARRAY\_UINT64**  
 lbmsdm.h, 651  
**LBMSDM\_TYPE\_ARRAY\_UINT8**  
 lbmsdm.h, 651  
**LBMSDM\_TYPE\_ARRAY\_UNICODE**  
 lbmsdm.h, 652  
**LBMSDM\_TYPE\_BLOB**  
 lbmsdm.h, 651  
**LBMSDM\_TYPE\_BOOLEAN**  
 lbmsdm.h, 650  
**LBMSDM\_TYPE\_DECIMAL**  
 lbmsdm.h, 651  
**LBMSDM\_TYPE\_DOUBLE**  
 lbmsdm.h, 651  
**LBMSDM\_TYPE\_FLOAT**  
 lbmsdm.h, 651  
**LBMSDM\_TYPE\_INT16**  
 lbmsdm.h, 650

LBMSDM\_TYPE\_INT32  
  lbmsdm.h, 651

LBMSDM\_TYPE\_INT64  
  lbmsdm.h, 651

LBMSDM\_TYPE\_INT8  
  lbmsdm.h, 650

LBMSDM\_TYPE\_INVALID  
  lbmsdm.h, 650

LBMSDM\_TYPE\_MESSAGE  
  lbmsdm.h, 651

LBMSDM\_TYPE\_STRING  
  lbmsdm.h, 651

LBMSDM\_TYPE\_TIMESTAMP  
  lbmsdm.h, 651

LBMSDM\_TYPE\_UINT16  
  lbmsdm.h, 650

LBMSDM\_TYPE\_UINT32  
  lbmsdm.h, 651

LBMSDM\_TYPE\_UINT64  
  lbmsdm.h, 651

LBMSDM\_TYPE\_UINT8  
  lbmsdm.h, 650

LBMSDM\_TYPE\_UNICODE  
  lbmsdm.h, 651

lbmsdm\_win32\_static\_init  
  lbmsdm.h, 669

lbtipc  
  lbm\_rcv\_transport\_stats\_t\_stct, 191  
  lbm\_src\_transport\_stats\_t\_stct, 234

lbtrdma  
  lbm\_rcv\_transport\_stats\_t\_stct, 191  
  lbm\_src\_transport\_stats\_t\_stct, 234

lbtrm  
  lbm\_rcv\_transport\_stats\_t\_stct, 191  
  lbm\_src\_transport\_stats\_t\_stct, 234

lbtrm\_unknown\_msgs\_reved  
  lbm\_context\_stats\_t\_stct, 127

lbtru  
  lbm\_rcv\_transport\_stats\_t\_stct, 191  
  lbm\_src\_transport\_stats\_t\_stct, 234

lbtru\_unknown\_msgs\_rcved  
  lbm\_context\_stats\_t\_stct, 127

len  
  lbm\_apphdr\_chain\_elem\_t\_stct, 113  
  lbm\_msg\_t\_stct, 156  
  lbmpdm\_field\_value\_stct\_t, 285

len\_arr  
  lbmpdm\_field\_value\_stct\_t, 286

lost  
  lbm\_rcv\_transport\_stats\_lbtrm\_t\_-  
    stct, 178

  lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
    stct, 185

low\_rxreq\_max\_sequence\_number  
  lbm\_ume\_rcv\_recovery\_info\_ex\_-  
    func\_info\_t\_stct, 244

low\_sequence\_number  
  lbm\_ume\_rcv\_recovery\_info\_ex\_-  
    func\_info\_t\_stct, 244

lu  
  lbm\_umq\_ulb\_application\_set\_-  
    attr\_t\_stct, 265

  lbm\_umq\_ulb\_receiver\_type\_attr\_-  
    t\_stct, 266

mant  
  lbmpdm\_decimal\_t, 283  
  lbmsdm\_decimal\_t\_stct, 288

mAttributeBlockLength  
  lbmmmon\_packet\_hdr\_t\_stct, 277

mc\_group  
  lbm\_transport\_source\_info\_t\_stct,  
    239

mCtxDeserialize  
  lbmmmon\_format\_func\_t\_stct, 276

mCtxSerialize  
  lbmmmon\_format\_func\_t\_stct, 276

mDataLength  
  lbmmmon\_packet\_hdr\_t\_stct, 277

mEntryCount  
  lbmmmon\_attr\_block\_t\_stct, 271

mEntryLength  
  lbmmmon\_attr\_block\_t\_stct, 271

mErrorMessage  
  lbmmmon\_format\_func\_t\_stct, 276  
  lbmmmon\_transport\_func\_t\_stct, 282

messages  
  lbm\_flight\_size\_inflight\_t\_stct, 144

mEvqDeserialize  
  lbmmmon\_format\_func\_t\_stct, 276

mEvqSerialize  
  lbmmmon\_format\_func\_t\_stct, 276

mFiller  
     lbmmmon\_packet\_hdr\_t\_stct, 277

mFinish  
     lbmmmon\_format\_func\_t\_stct, 276

mFinishReceiver  
     lbmmmon\_transport\_func\_t\_stct, 281

mFinishSource  
     lbmmmon\_transport\_func\_t\_stct, 281

mInit  
     lbmmmon\_format\_func\_t\_stct, 275

mInitReceiver  
     lbmmmon\_transport\_func\_t\_stct, 281

mInitSource  
     lbmmmon\_transport\_func\_t\_stct, 281

mLength  
     lbmmmon\_attr\_entry\_t\_stct, 272

mRcvDeserialize  
     lbmmmon\_format\_func\_t\_stct, 275

mRcvSerialize  
     lbmmmon\_format\_func\_t\_stct, 275

mReceive  
     lbmmmon\_transport\_func\_t\_stct, 281

mSend  
     lbmmmon\_transport\_func\_t\_stct, 281

msg  
     lbm\_umq\_queue\_msg\_status\_t, 261

msg\_clientid  
     lbum\_src\_event\_sequence\_number\_info\_t\_stct, 199

    lbum\_src\_event\_ume\_ack\_ex\_info\_t\_stct, 202

    lbum\_src\_event\_ume\_ack\_info\_t\_stct, 203

    lbum\_src\_event\_umq\_message\_id\_info\_t\_stct, 210

    lbum\_src\_event\_umq\_stability\_ack\_info\_ex\_t\_stct, 214

    lbum\_src\_event\_umq\_ulb\_message\_info\_ex\_t\_stct, 216

msg\_id  
     lbum\_src\_event\_umq\_message\_id\_info\_t\_stct, 210

    lbum\_src\_event\_umq\_stability\_ack\_info\_ex\_t\_stct, 213

    lbum\_src\_event\_umq\_ulb\_message\_info\_ex\_t\_stct, 215

msgid  
     lbum\_umq\_queue\_msg\_status\_t, 261

msgs  
     lbum\_rcv\_umq\_queue\_msg\_list\_info\_t, 194

    lbum\_rcv\_umq\_queue\_msg\_retrieve\_info\_t, 195

msgs\_rcved  
     lbum\_rcv\_transport\_stats\_lbtipc\_t\_stct, 173

    lbum\_rcv\_transport\_stats\_lbtrdma\_t\_stct, 175

    lbum\_rcv\_transport\_stats\_lbtrm\_t\_stct, 178

    lbum\_rcv\_transport\_stats\_lbtru\_t\_stct, 185

msgs\_sent  
     lbum\_src\_transport\_stats\_lbtipc\_t\_stct, 224

    lbum\_src\_transport\_stats\_lbtrdma\_t\_stct, 225

    lbum\_src\_transport\_stats\_lbtrm\_t\_stct, 226

    lbum\_src\_transport\_stats\_lbtru\_t\_stct, 230

mSignature  
     lbmmmon\_packet\_hdr\_t\_stct, 277

mSrcDeserialize  
     lbmmmon\_format\_func\_t\_stct, 275

mSrcSerialize  
     lbmmmon\_format\_func\_t\_stct, 275

mType  
     lbmmmon\_attr\_entry\_t\_stct, 272

    lbmmmon\_packet\_hdr\_t\_stct, 277

nak\_pkts\_rcved  
     lbum\_src\_transport\_stats\_lbtrm\_t\_stct, 227

    lbum\_src\_transport\_stats\_lbtru\_t\_stct, 230

nak\_pkts\_sent  
     lbum\_rcv\_transport\_stats\_lbtrm\_t\_stct, 178

    lbum\_rcv\_transport\_stats\_lbtru\_t\_stct, 185

nak\_stm\_max

lbm\_rcv\_transport\_stats\_lbtrm\_t\_-  
stct, 180  
lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
stct, 187  
nak\_stm\_mean  
  lbm\_rcv\_transport\_stats\_lbtrm\_t\_-  
    stct, 179  
  lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
    stct, 186  
nak\_stm\_min  
  lbm\_rcv\_transport\_stats\_lbtrm\_t\_-  
    stct, 179  
  lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
    stct, 186  
nak\_tx\_max  
  lbm\_rcv\_transport\_stats\_lbtrm\_t\_-  
    stct, 180  
  lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
    stct, 187  
nak\_tx\_mean  
  lbm\_rcv\_transport\_stats\_lbtrm\_t\_-  
    stct, 180  
  lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
    stct, 187  
nak\_tx\_min  
  lbm\_rcv\_transport\_stats\_lbtrm\_t\_-  
    stct, 180  
  lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
    stct, 187  
naks\_ignored  
  lbm\_src\_transport\_stats\_lbtrm\_t\_-  
    stct, 227  
  lbm\_src\_transport\_stats\_lbtru\_t\_-  
    stct, 231  
naks\_rcved  
  lbm\_src\_transport\_stats\_lbtrm\_t\_-  
    stct, 227  
  lbm\_src\_transport\_stats\_lbtru\_t\_-  
    stct, 230  
naks\_rx\_delay\_ignored  
  lbm\_src\_transport\_stats\_lbtrm\_t\_-  
    stct, 227  
  lbm\_src\_transport\_stats\_lbtru\_t\_-  
    stct, 231  
naks\_sent  
lbm\_rcv\_transport\_stats\_lbtrm\_t\_-  
stct, 178  
lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
stct, 185  
naks\_shed  
  lbm\_src\_transport\_stats\_lbtrm\_t\_-  
    stct, 227  
  lbm\_src\_transport\_stats\_lbtru\_t\_-  
    stct, 231  
name  
  lbm\_ume\_store\_name\_entry\_t\_stct,  
    254  
  lbm\_umq\_queue\_entry\_t\_stct, 260  
ncfs\_ignored  
  lbm\_rcv\_transport\_stats\_lbtrm\_t\_-  
    stct, 178  
  lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
    stct, 185  
ncfs\_rx\_delay  
  lbm\_rcv\_transport\_stats\_lbtrm\_t\_-  
    stct, 179  
  lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
    stct, 186  
ncfs\_shed  
  lbm\_rcv\_transport\_stats\_lbtrm\_t\_-  
    stct, 179  
  lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
    stct, 186  
ncfs\_unknown  
  lbm\_rcv\_transport\_stats\_lbtrm\_t\_-  
    stct, 179  
  lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
    stct, 186  
num\_appsets  
  lbm\_umq\_queue\_topic\_t\_stct, 264  
num\_arr\_elem  
  lbmpdm\_field\_info\_attr\_stct\_t, 284  
  lbmpdm\_field\_value\_stct\_t, 285  
num\_clients  
  lbm\_src\_transport\_stats\_lbtrdma\_t\_-  
    stct, 224  
  lbm\_src\_transport\_stats\_lbtrdma\_t\_-  
    stct, 225  
  lbm\_src\_transport\_stats\_lbtru\_t\_-  
    stct, 232

<b>lbtm_src_transport_stats_tcp_t_stct,</b>	PDM_FAILURE
235	lbmpdm.h, 591
<b>num_msgs</b>	PDM_FALSE
<b>lbtm_rcv_umq_queue_msg_list_</b>	lbmpdm.h, 591
info_t, 194	PDM_FIELD_INFO_FLAG_FIXED_-
<b>lbtm_rcv_umq_queue_msg_</b>	STR_LEN
retrieve_info_t, 195	lbmpdm.h, 591
<b>num_topics</b>	PDM_FIELD_INFO_FLAG_NUM_-
<b>lbtm_ctx_umq_queue_topic_list_</b>	ARR_ELEM
info_t, 129	lbmpdm.h, 591
<b>offset</b>	PDM_FIELD_INFO_FLAG_REQ
<b>lbtm_msg_fragment_info_t_stct</b> , 150	lbmpdm.h, 591
<b>out_of_order</b>	PDM_INTERNAL_TYPE_INVALID
<b>lbtm_rcv_transport_stats_lbtrm_t_</b>	lbmpdm.h, 591
stct, 182	PDM_ITER_INVALID_FIELD_-
<b>password</b>	HANDLE
<b>lbtm_umm_info_t_stct</b> , 255	lbmpdm.h, 591
<b>PDM_DEFN_INT_FIELD_NAMES</b>	PDM_MSG_FLAG_DEL_DEFN_-
lbmpdm.h, 589	WHEN_REPLACED
<b>PDM_DEFN_STR_FIELD_NAMES</b>	lbmpdm.h, 591
lbmpdm.h, 589	PDM_MSG_FLAG_INCL_DEFN
<b>PDM_ERR_CREATE_BUFFER</b>	lbmpdm.h, 591
lbmpdm.h, 589	PDM_MSG_FLAG_NEED_BYTE_-
<b>PDM_ERR_CREATE_SECTION</b>	SWAP
lbmpdm.h, 590	lbmpdm.h, 592
<b>PDM_ERR_DEFN_INVALID</b>	PDM_MSG_FLAG_TRY_LOAD_-
lbmpdm.h, 590	DEFN_FROM_CACHE
<b>PDM_ERR_EINVAL</b>	lbmpdm.h, 592
lbmpdm.h, 590	PDM_MSG_FLAG_USE_MSG_-
<b>PDM_ERR_FIELD_IS_NULL</b>	DEFN_IF_NEEDED
lbmpdm.h, 590	lbmpdm.h, 592
<b>PDM_ERR_FIELD_NOT_FOUND</b>	PDM_MSG_FLAG_VAR_OR_OPT_-
lbmpdm.h, 590	FLDS_SET
<b>PDM_ERR_INSUFFICIENT_-</b>	lbmpdm.h, 592
BUFFER_LENGTH	PDM_MSG_VER_POLICY_BEST
lbmpdm.h, 590	lbmpdm.h, 592
<b>PDM_ERR_MSG_INVALID</b>	PDM_MSG_VER_POLICY_EXACT
lbmpdm.h, 590	lbmpdm.h, 592
<b>PDM_ERR_NO_MORE_FIELDS</b>	PDM_SUCCESS
lbmpdm.h, 590	lbmpdm.h, 592
<b>PDM_ERR_NOMEM</b>	PDM_TRUE
lbmpdm.h, 590	lbmpdm.h, 592
<b>PDM_ERR_REQ_FIELD_NOT_SET</b>	PDM_TYPE_BLOB
lbmpdm.h, 590	lbmpdm.h, 592
	PDM_TYPE_BLOB_ARR
	lbmpdm.h, 593

PDM\_TYPE\_BOOLEAN  
lbmpdm.h, 593

PDM\_TYPE\_BOOLEAN\_ARR  
lbmpdm.h, 593

PDM\_TYPE\_DECIMAL  
lbmpdm.h, 593

PDM\_TYPE\_DECIMAL\_ARR  
lbmpdm.h, 593

PDM\_TYPE\_DOUBLE  
lbmpdm.h, 593

PDM\_TYPE\_DOUBLE\_ARR  
lbmpdm.h, 593

PDM\_TYPE\_FIX\_STRING  
lbmpdm.h, 593

PDM\_TYPE\_FIX\_STRING\_ARR  
lbmpdm.h, 593

PDM\_TYPE\_FIX\_UNICODE  
lbmpdm.h, 593

PDM\_TYPE\_FIX\_UNICODE\_ARR  
lbmpdm.h, 594

PDM\_TYPE\_FLOAT  
lbmpdm.h, 594

PDM\_TYPE\_FLOAT\_ARR  
lbmpdm.h, 594

PDM\_TYPE\_INT16  
lbmpdm.h, 594

PDM\_TYPE\_INT16\_ARR  
lbmpdm.h, 594

PDM\_TYPE\_INT32  
lbmpdm.h, 594

PDM\_TYPE\_INT32\_ARR  
lbmpdm.h, 594

PDM\_TYPE\_INT64  
lbmpdm.h, 594

PDM\_TYPE\_INT64\_ARR  
lbmpdm.h, 594

PDM\_TYPE\_INT8  
lbmpdm.h, 594

PDM\_TYPE\_INT8\_ARR  
lbmpdm.h, 595

PDM\_TYPE\_MESSAGE  
lbmpdm.h, 595

PDM\_TYPE\_MESSAGE\_ARR  
lbmpdm.h, 595

PDM\_TYPE\_STRING  
lbmpdm.h, 595

PDM\_TYPE\_STRING\_ARR  
lbmpdm.h, 595

PDM\_TYPE\_TIMESTAMP  
lbmpdm.h, 595

PDM\_TYPE\_TIMESTAMP\_ARR  
lbmpdm.h, 595

PDM\_TYPE\_UINT16  
lbmpdm.h, 595

PDM\_TYPE\_UINT16\_ARR  
lbmpdm.h, 595

PDM\_TYPE\_UINT32  
lbmpdm.h, 595

PDM\_TYPE\_UINT32\_ARR  
lbmpdm.h, 596

PDM\_TYPE\_UINT64  
lbmpdm.h, 596

PDM\_TYPE\_UINT64\_ARR  
lbmpdm.h, 596

PDM\_TYPE\_UINT8  
lbmpdm.h, 596

PDM\_TYPE\_UINT8\_ARR  
lbmpdm.h, 596

PDM\_TYPE\_UNICODE  
lbmpdm.h, 596

PDM\_TYPE\_UNICODE\_ARR  
lbmpdm.h, 596

properties  
  lbm\_msg\_t\_stct, 157  
  lbm\_src\_send\_ex\_info\_t\_stct, 222

queue  
  lbm\_context\_event\_umq\_-  
    registration\_complete\_ex\_-  
    t\_stct, 119  
  lbm\_context\_event\_umq\_-  
    registration\_ex\_t\_stct, 122  
  lbm\_msg\_umq\_registration\_-  
    complete\_ex\_t\_stct, 169  
  lbm\_src\_event\_umq\_registration\_-  
    complete\_ex\_t\_stct, 212  
  lbm\_src\_event\_umq\_stability\_ack\_-  
    info\_ex\_t\_stct, 214

queue\_id  
  lbm\_context\_event\_umq\_-  
    registration\_complete\_ex\_-  
    t\_stct, 119

**l**  
 lbm\_context\_event\_umq\_-  
     registration\_ex\_t\_stct, 121  
 lbm\_msg\_umq\_deregistration\_-  
     complete\_ex\_t\_stct, 164  
 lbm\_msg\_umq\_index\_assigned\_-  
     ex\_t\_stct, 165  
 lbm\_msg\_umq\_index\_assignment\_-  
     eligibility\_start\_complete\_ex\_-  
     t\_stct, 166  
 lbm\_msg\_umq\_index\_assignment\_-  
     eligibility\_stop\_complete\_ex\_-  
     t\_stct, 167  
 lbm\_msg\_umq\_index\_released\_ex\_-  
     t\_stct, 168  
 lbm\_msg\_umq\_registration\_-  
     complete\_ex\_t\_stct, 169  
 lbm\_src\_event\_umq\_registration\_-  
     complete\_ex\_t\_stct, 212  
 lbm\_src\_event\_umq\_stability\_ack\_-  
     info\_ex\_t\_stct, 214  
 queue\_instance  
     lbm\_context\_event\_umq\_-  
         registration\_ex\_t\_stct, 121  
     lbm\_src\_event\_umq\_stability\_ack\_-  
         info\_ex\_t\_stct, 214  
 queue\_instance\_index  
     lbm\_context\_event\_umq\_-  
         registration\_ex\_t\_stct, 121  
     lbm\_src\_event\_umq\_stability\_ack\_-  
         info\_ex\_t\_stct, 214  
 rctlr\_data\_msgs  
     lbm\_src\_transport\_stats\_lbtrm\_t\_-  
         stct, 228  
 rctlr\_rx\_msgs  
     lbm\_src\_transport\_stats\_lbtrm\_t\_-  
         stct, 228  
 rcv\_registration\_id  
     lbm\_msg\_ume\_deregistration\_ex\_-  
         t\_stct, 158  
     lbm\_msg\_ume\_registration\_ex\_t\_-  
         stct, 161  
     lbm\_msg\_ume\_registration\_t\_stct,  
         163  
     lbm\_src\_event\_ume\_ack\_ex\_info\_-  
         t\_stct, 201  
 lbm\_src\_event\_ume\_ack\_info\_t\_-  
     stct, 203  
 receiver  
     lbm\_src\_event\_umq\_ulb\_message\_-  
         info\_ex\_t\_stct, 216  
     lbm\_src\_event\_umq\_ulb\_receiver\_-  
         info\_ex\_t\_stct, 217  
 regid  
     lbm\_umqmsgid\_t\_stct, 259  
     lbm\_umq\_queue\_entry\_t\_stct, 260  
 registration\_id  
     lbm\_context\_event\_umq\_-  
         registration\_complete\_ex\_-  
         t\_stct, 119  
     lbm\_context\_event\_umq\_-  
         registration\_ex\_t\_stct, 121  
     lbm\_src\_event\_ume\_-  
         deregistration\_ex\_t\_stct,  
         204  
     lbm\_src\_event\_ume\_registration\_-  
         ex\_t\_stct, 207  
     lbm\_src\_event\_ume\_registration\_t\_-  
         stct, 209  
     lbm\_src\_event\_umq\_ulb\_message\_-  
         info\_ex\_t\_stct, 215  
     lbm\_src\_event\_umq\_ulb\_receiver\_-  
         info\_ex\_t\_stct, 217  
     lbm\_ume\_store\_entry\_t\_stct, 252  
     lbm\_ume\_store\_name\_entry\_t\_stct,  
         254  
 req  
     lbmpdm\_field\_info\_attr\_stct\_t, 284  
 reserved  
     lbm\_umq\_queue\_topic\_t\_stct, 264  
 resolver\_ip  
     lbm\_icast\_resolver\_entry\_t\_stct,  
         241  
 resp\_blocked  
     lbm\_context\_stats\_t\_stct, 127  
 resp\_msgs  
     lbm\_event\_queue\_stats\_t\_stct, 134  
 resp\_msgs\_svc\_max  
     lbm\_event\_queue\_stats\_t\_stct, 135  
 resp\_msgs\_svc\_mean  
     lbm\_event\_queue\_stats\_t\_stct, 134  
 resp\_msgs\_svc\_min

lbm_event_queue_stats_t_stct, 134	lbm_transport_source_info_t_stct, 239
resp_msgs_tot	Set a field value in a message by field index, 70
lbm_event_queue_stats_t_stct, 134	Set a field value in a message by field index to an array field, 85
resp_would_block	Set a field value in a message by field name, 75
lbm_context_stats_t_stct, 128	Set a field value in a message by field name to an array field, 89
response	Set a field value in a message referenced by an iterator, 80
lbm_msg_t_stct, 156	Set a field value in a message, referenced by an iterator, to an array field., 93
rx_bytes_sent	Set an array field element value by field index, 97
lbm_src_transport_stats_lbtrm_t_- stct, 228	Set an array field element value by field name, 102
lbm_src_transport_stats_lbtru_t_- stct, 232	Set an array field element value for a field referenced by an iterator, 107
rxs_sent	set_array_idx
lbm_src_transport_stats_lbtrm_t_- stct, 228	lbmsdm_msg_set_blob_array_idx, 86
lbm_src_transport_stats_lbtru_t_- stct, 231	lbmsdm_msg_set_boolean_array_- idx, 86
send_blocked	lbmsdm_msg_set_decimal_array_- idx, 86
lbm_context_stats_t_stct, 127	lbmsdm_msg_set_double_array_- idx, 86
send_would_block	lbmsdm_msg_set_float_array_idx, 86
lbm_context_stats_t_stct, 127	lbmsdm_msg_set_int16_array_idx, 86
sequence_number	lbmsdm_msg_set_int32_array_idx, 87
lbm_msg_gateway_info_t_stct, 151	lbmsdm_msg_set_int64_array_idx, 87
lbm_msg_t_stct, 156	lbmsdm_msg_set_int8_array_idx, 87
lbm_msg_ume_deregistration_ex_- t_stct, 158	lbmsdm_msg_set_message_array_- idx, 87
lbm_msg_ume_registration_- complete_ex_t_stct, 160	lbmsdm_msg_set_string_array_idx, 87
lbm_msg_ume_registration_ex_t_- stct, 161	lbmsdm_msg_set_timestamp_- array_idx, 87
lbm_src_event_ume_ack_ex_info_- t_stct, 201	
lbm_src_event_ume_ack_info_t_- stct, 203	
lbm_src_event_ume_- deregistration_ex_t_stct, 204	
lbm_src_event_ume_registration_- complete_ex_t_stct, 206	
lbm_src_event_ume_registration_- ex_t_stct, 207	
servers	
lbm_umm_info_t_stct, 255	
session_id	

lbmsdm\_msg\_set\_uint16\_array\_idx,  
     87  
 lbmsdm\_msg\_set\_uint32\_array\_idx,  
     88  
 lbmsdm\_msg\_set\_uint64\_array\_idx,  
     88  
 lbmsdm\_msg\_set\_uint8\_array\_idx,  
     88  
 lbmsdm\_msg\_set\_unicode\_array\_idx,  
     88  
**set\_array\_iter**  
     lbmsdm\_iter\_set\_blob\_array, 93  
     lbmsdm\_iter\_set\_boolean\_array, 93  
     lbmsdm\_iter\_set\_decimal\_array, 94  
     lbmsdm\_iter\_set\_double\_array, 94  
     lbmsdm\_iter\_set\_float\_array, 94  
     lbmsdm\_iter\_set\_int16\_array, 94  
     lbmsdm\_iter\_set\_int32\_array, 94  
     lbmsdm\_iter\_set\_int64\_array, 94  
     lbmsdm\_iter\_set\_int8\_array, 95  
     lbmsdm\_iter\_set\_message\_array, 95  
     lbmsdm\_iter\_set\_string\_array, 95  
     lbmsdm\_iter\_set\_timestamp\_array,  
         95  
     lbmsdm\_iter\_set\_uint16\_array, 95  
     lbmsdm\_iter\_set\_uint32\_array, 95  
     lbmsdm\_iter\_set\_uint64\_array, 95  
     lbmsdm\_iter\_set\_uint8\_array, 96  
     lbmsdm\_iter\_set\_unicode\_array, 96  
**set\_array\_name**  
     lbmsdm\_msg\_set\_blob\_array\_name,  
         90  
     lbmsdm\_msg\_set\_boolean\_array\_-  
         name, 90  
     lbmsdm\_msg\_set\_decimal\_array\_-  
         name, 90  
     lbmsdm\_msg\_set\_double\_array\_-  
         name, 90  
     lbmsdm\_msg\_set\_float\_array\_name,  
         90  
     lbmsdm\_msg\_set\_int16\_array\_-  
         name, 90  
     lbmsdm\_msg\_set\_int32\_array\_-  
         name, 91  
     lbmsdm\_msg\_set\_int64\_array\_-  
         name, 91  
     lbmsdm\_msg\_set\_int8\_array\_name,  
         91  
     lbmsdm\_msg\_set\_message\_array\_-  
         name, 91  
     lbmsdm\_msg\_set\_string\_array\_-  
         name, 91  
     lbmsdm\_msg\_set\_timestamp\_-  
         array\_name, 91  
     lbmsdm\_msg\_set\_uint16\_array\_-  
         name, 91  
     lbmsdm\_msg\_set\_uint32\_array\_-  
         name, 92  
     lbmsdm\_msg\_set\_uint64\_array\_-  
         name, 92  
     lbmsdm\_msg\_set\_uint8\_array\_-  
         name, 92  
     lbmsdm\_msg\_set\_unicode\_array\_-  
         name, 92  
**set\_elem\_idx**  
     lbmsdm\_msg\_set\_blob\_elem\_idx,  
         98  
     lbmsdm\_msg\_set\_boolean\_elem\_-  
         idx, 98  
     lbmsdm\_msg\_set\_decimal\_elem\_-  
         idx, 98  
     lbmsdm\_msg\_set\_double\_elem\_idx,  
         98  
     lbmsdm\_msg\_set\_float\_elem\_idx,  
         99  
     lbmsdm\_msg\_set\_int16\_elem\_idx,  
         99  
     lbmsdm\_msg\_set\_int32\_elem\_idx,  
         99  
     lbmsdm\_msg\_set\_int64\_elem\_idx,  
         99  
     lbmsdm\_msg\_set\_int8\_elem\_idx, 99  
     lbmsdm\_msg\_set\_message\_elem\_-  
         idx, 99  
     lbmsdm\_msg\_set\_string\_elem\_idx,  
         99  
     lbmsdm\_msg\_set\_timestamp\_-  
         elem\_idx, 100  
     lbmsdm\_msg\_set\_uint16\_elem\_idx,  
         100  
     lbmsdm\_msg\_set\_uint32\_elem\_idx,  
         100

lbmsdm\_msg\_set\_uint64\_elem\_idx,  
    100  
lbmsdm\_msg\_set\_uint8\_elem\_idx,  
    100  
lbmsdm\_msg\_set\_unicode\_elem\_ -  
    idx, 100  
set\_elem\_iter  
    lbmsdm\_iter\_set\_blob\_elem, 108  
    lbmsdm\_iter\_set\_boolean\_elem, 108  
    lbmsdm\_iter\_set\_decimal\_elem, 108  
    lbmsdm\_iter\_set\_double\_elem, 108  
    lbmsdm\_iter\_set\_float\_elem, 109  
    lbmsdm\_iter\_set\_int16\_elem, 109  
    lbmsdm\_iter\_set\_int32\_elem, 109  
    lbmsdm\_iter\_set\_int64\_elem, 109  
    lbmsdm\_iter\_set\_int8\_elem, 109  
    lbmsdm\_iter\_set\_message\_elem,  
        109  
    lbmsdm\_iter\_set\_string\_elem, 109  
    lbmsdm\_iter\_set\_timestamp\_elem,  
        110  
    lbmsdm\_iter\_set\_uint16\_elem, 110  
    lbmsdm\_iter\_set\_uint32\_elem, 110  
    lbmsdm\_iter\_set\_uint64\_elem, 110  
    lbmsdm\_iter\_set\_uint8\_elem, 110  
    lbmsdm\_iter\_set\_unicode\_elem, 110  
set\_elem\_name  
    lbmsdm\_msg\_set\_blob\_elem\_name,  
        103  
    lbmsdm\_msg\_set\_boolean\_elem\_ -  
        name, 103  
    lbmsdm\_msg\_set\_decimal\_elem\_ -  
        name, 103  
    lbmsdm\_msg\_set\_double\_elem\_ -  
        name, 104  
    lbmsdm\_msg\_set\_float\_elem\_name,  
        104  
    lbmsdm\_msg\_set\_int16\_elem\_ -  
        name, 104  
    lbmsdm\_msg\_set\_int32\_elem\_ -  
        name, 104  
    lbmsdm\_msg\_set\_int64\_elem\_ -  
        name, 104  
    lbmsdm\_msg\_set\_int8\_elem\_name,  
        104  
lbmsdm\_msg\_set\_message\_elem\_ -  
    name, 105  
lbmsdm\_msg\_set\_string\_elem\_ -  
    name, 105  
lbmsdm\_msg\_set\_timestamp\_ -  
    elem\_name, 105  
lbmsdm\_msg\_set\_uint16\_elem\_ -  
    name, 105  
lbmsdm\_msg\_set\_uint32\_elem\_ -  
    name, 105  
lbmsdm\_msg\_set\_uint64\_elem\_ -  
    name, 105  
lbmsdm\_msg\_set\_uint8\_elem\_ -  
    name, 106  
lbmsdm\_msg\_set\_unicode\_elem\_ -  
    name, 106  
set\_idx  
    lbmsdm\_msg\_set\_blob\_idx, 71  
    lbmsdm\_msg\_set\_boolean\_idx, 71  
    lbmsdm\_msg\_set\_decimal\_idx, 71  
    lbmsdm\_msg\_set\_double\_idx, 71  
    lbmsdm\_msg\_set\_float\_idx, 72  
    lbmsdm\_msg\_set\_int16\_idx, 72  
    lbmsdm\_msg\_set\_int32\_idx, 72  
    lbmsdm\_msg\_set\_int64\_idx, 72  
    lbmsdm\_msg\_set\_int8\_idx, 72  
    lbmsdm\_msg\_set\_message\_idx, 72  
    lbmsdm\_msg\_set\_string\_idx, 72  
    lbmsdm\_msg\_set\_timestamp\_idx,  
        73  
lbmsdm\_msg\_set\_uint16\_idx, 73  
lbmsdm\_msg\_set\_uint32\_idx, 73  
lbmsdm\_msg\_set\_uint64\_idx, 73  
lbmsdm\_msg\_set\_uint8\_idx, 73  
lbmsdm\_msg\_set\_unicode\_idx, 73  
set\_iter  
    lbmsdm\_iter\_set\_blob, 81  
    lbmsdm\_iter\_set\_boolean, 81  
    lbmsdm\_iter\_set\_decimal, 81  
    lbmsdm\_iter\_set\_double, 81  
    lbmsdm\_iter\_set\_float, 82  
    lbmsdm\_iter\_set\_int16, 82  
    lbmsdm\_iter\_set\_int32, 82  
    lbmsdm\_iter\_set\_int64, 82  
    lbmsdm\_iter\_set\_int8, 82  
    lbmsdm\_iter\_set\_message, 82

**lbmsdm\_iter\_set\_string**, 82  
**lbmsdm\_iter\_set\_timestamp**, 83  
**lbmsdm\_iter\_set\_uint16**, 83  
**lbmsdm\_iter\_set\_uint32**, 83  
**lbmsdm\_iter\_set\_uint64**, 83  
**lbmsdm\_iter\_set\_uint8**, 83  
**lbmsdm\_iter\_set\_unicode**, 83  
**set\_name**  
    **lbmsdm\_msg\_set\_blob\_name**, 76  
    **lbmsdm\_msg\_set\_boolean\_name**,  
        76  
    **lbmsdm\_msg\_set\_decimal\_name**,  
        76  
    **lbmsdm\_msg\_set\_double\_name**, 76  
    **lbmsdm\_msg\_set\_float\_name**, 77  
    **lbmsdm\_msg\_set\_int16\_name**, 77  
    **lbmsdm\_msg\_set\_int32\_name**, 77  
    **lbmsdm\_msg\_set\_int64\_name**, 77  
    **lbmsdm\_msg\_set\_int8\_name**, 77  
    **lbmsdm\_msg\_set\_message\_name**,  
        77  
    **lbmsdm\_msg\_set\_string\_name**, 77  
    **lbmsdm\_msg\_set\_timestamp\_name**,  
        78  
    **lbmsdm\_msg\_set\_uint16\_name**, 78  
    **lbmsdm\_msg\_set\_uint32\_name**, 78  
    **lbmsdm\_msg\_set\_uint64\_name**, 78  
    **lbmsdm\_msg\_set\_uint8\_name**, 78  
    **lbmsdm\_msg\_set\_unicode\_name**,  
        78  
**SLEEP\_MSEC**  
    **umeblocksr.h**, 671  
**source**  
    **lbm\_msg\_gateway\_info\_t\_stct**, 151  
    **lbm\_msg\_t\_stct**, 154  
    **lbm\_rcv\_transport\_stats\_t\_stct**, 190  
    **lbm\_src\_transport\_stats\_t\_stct**, 233  
    **lbm\_ume\_rcv\_recovery\_info\_ex\_-**  
        **func\_info\_t\_stct**, 245  
    **lbm\_ume\_rcv\_regid\_ex\_func\_info\_-**  
        **t\_stct**, 248  
**source\_clientd**  
    **lbm\_msg\_t\_stct**, 156  
    **lbm\_ume\_rcv\_recovery\_info\_ex\_-**  
        **func\_info\_t\_stct**, 245  
**lbm\_ume\_rcgid\_ex\_func\_info\_-**  
    **t\_stct**, 247  
**source\_events**  
    **lbm\_event\_queue\_stats\_t\_stct**, 138  
**source\_events\_svc\_max**  
    **lbm\_event\_queue\_stats\_t\_stct**, 139  
**source\_events\_svc\_mean**  
    **lbm\_event\_queue\_stats\_t\_stct**, 139  
**source\_events\_svc\_min**  
    **lbm\_event\_queue\_stats\_t\_stct**, 139  
**source\_events\_tot**  
    **lbm\_event\_queue\_stats\_t\_stct**, 138  
**source\_port**  
    **lbm\_uicast\_resolver\_entry\_t\_stct**,  
        241  
**src\_ip**  
    **lbm\_transport\_source\_info\_t\_stct**,  
        239  
**src\_port**  
    **lbm\_transport\_source\_info\_t\_stct**,  
        239  
**src\_registration\_id**  
    **lbm\_ume\_deregistration\_ex\_-**  
        **t\_stct**, 158  
    **lbm\_ume\_registration\_ex\_t\_-**  
        **stct**, 161  
    **lbm\_ume\_registration\_t\_stct**,  
        163  
    **lbm\_ume\_rcv\_regid\_ex\_func\_info\_-**  
        **t\_stct**, 247  
**stamp**  
    **lbm\_umqmsgid\_t\_stct**, 259  
**start\_sequence\_number**  
    **lbm\_msg\_fragment\_info\_t\_stct**, 150  
**state**  
    **lbm\_src\_event\_flight\_size\_-**  
        **notification\_t\_stct**, 198  
**status**  
    **lbm\_async\_operation\_info\_t**, 117  
    **lbm\_umqqueue\_msg\_status\_t**, 261  
    **lbm\_umqqueue\_topic\_status\_t**, 263  
**store**  
    **lbm\_ume\_deregistration\_ex\_-**  
        **t\_stct**, 159  
    **lbm\_ume\_registration\_ex\_t\_-**  
        **stct**, 162

lbm\_src\_event\_ume\_ack\_ex\_info\_t\_stct, 201  
lbm\_src\_event\_ume\_deregistration\_ex\_t\_stct, 205  
lbm\_src\_event\_ume\_registration\_ex\_t\_stct, 207  
lbm\_ume\_rcv\_regid\_ex\_func\_info\_t\_stct, 248  
store\_index  
    lbm\_msg\_ume\_deregistration\_ex\_t\_stct, 159  
    lbm\_msg\_ume\_registration\_ex\_t\_stct, 161  
    lbm\_src\_event\_ume\_ack\_ex\_info\_t\_stct, 202  
    lbm\_src\_event\_ume\_deregistration\_ex\_t\_stct, 204  
    lbm\_src\_event\_ume\_registration\_ex\_t\_stct, 207  
    lbm\_ume\_rcv\_regid\_ex\_func\_info\_t\_stct, 247  
subtype  
    lbm\_apphdr\_chain\_elem\_t\_stct, 113  
tcp  
    lbm\_rcv\_transport\_stats\_t\_stct, 191  
    lbm\_src\_transport\_stats\_t\_stct, 234  
tcp\_port  
    lbm\_ume\_store\_entry\_t\_stct, 252  
timer\_events  
    lbm\_event\_queue\_stats\_t\_stct, 137  
timer\_events\_svc\_max  
    lbm\_event\_queue\_stats\_t\_stct, 138  
timer\_events\_svc\_mean  
    lbm\_event\_queue\_stats\_t\_stct, 138  
timer\_events\_svc\_min  
    lbm\_event\_queue\_stats\_t\_stct, 138  
timer\_events\_tot  
    lbm\_event\_queue\_stats\_t\_stct, 138  
topic  
    lbm\_umq\_queue\_topic\_status\_t, 263  
topic\_idx  
    lbm\_transport\_source\_info\_t\_stct, 239  
topic\_name  
    lbm\_msg\_t\_stct, 154  
    lbm\_umq\_queue\_topic\_t\_stct, 264  
topicless\_im\_msgs  
    lbm\_event\_queue\_stats\_t\_stct, 135  
topicless\_im\_msgs\_svc\_max  
    lbm\_event\_queue\_stats\_t\_stct, 136  
topicless\_im\_msgs\_svc\_mean  
    lbm\_event\_queue\_stats\_t\_stct, 135  
topicless\_im\_msgs\_svc\_min  
    lbm\_event\_queue\_stats\_t\_stct, 135  
topicless\_im\_msgs\_tot  
    lbm\_event\_queue\_stats\_t\_stct, 135  
topics  
    lbm\_ctx\_umq\_queue\_topic\_list\_info\_t, 129  
total\_message\_length  
    lbm\_msg\_fragment\_info\_t\_stct, 150  
tr\_bytes\_rcved  
    lbm\_context\_stats\_t\_stct, 126  
tr\_bytes\_sent  
    lbm\_context\_stats\_t\_stct, 125  
tr\_dgrams\_dropped\_malformed  
    lbm\_context\_stats\_t\_stct, 126  
tr\_dgrams\_dropped\_type  
    lbm\_context\_stats\_t\_stct, 126  
tr\_dgrams\_dropped\_ver  
    lbm\_context\_stats\_t\_stct, 126  
tr\_dgrams\_rcved  
    lbm\_context\_stats\_t\_stct, 126  
tr\_dgrams\_send\_failed  
    lbm\_context\_stats\_t\_stct, 126  
tr\_dgrams\_sent  
    lbm\_context\_stats\_t\_stct, 125  
tr\_rcv\_topics  
    lbm\_context\_stats\_t\_stct, 127  
tr\_rcv\_unresolved\_topics  
    lbm\_context\_stats\_t\_stct, 127  
tr\_src\_topics  
    lbm\_context\_stats\_t\_stct, 126  
transport\_id  
    lbm\_transport\_source\_info\_t\_stct, 239  
tv\_secs  
    lbmpdm\_timestamp\_t, 287  
tv\_usecs

lbmpdm\_timestamp\_t, 287  
 txw\_bytes  
     lbm\_src\_transport\_stats\_lbtrm\_t\_stct, 227  
 txw\_msgs  
     lbm\_src\_transport\_stats\_lbtrm\_t\_stct, 226  
 type  
     lmb\_apphdr\_chain\_elem\_t\_stct, 113  
     lmb\_async\_operation\_info\_t, 117  
     lmb\_msg\_t\_stct, 154  
     lmb\_rcv\_transport\_stats\_t\_stct, 190  
     lmb\_src\_event\_flight\_size\_notification\_t\_stct, 198  
     lmb\_src\_transport\_stats\_t\_stct, 233  
     lmb\_transport\_source\_info\_t\_stct, 239  
  
 u32  
     lmb\_hf\_sequence\_number\_t\_stct, 145  
  
 u64  
     lmb\_hf\_sequence\_number\_t\_stct, 145  
  
 uim\_dup\_msgs\_rcved  
     lmb\_context\_stats\_t\_stct, 128  
  
 uim\_msgs\_no\_stream\_rcved  
     lmb\_context\_stats\_t\_stct, 128  
  
 UME\_BLOCK\_DEBUG\_PRINT  
     umeblocks.h, 671  
  
 UME\_BLOCK\_PRINT\_ERROR  
     umeblocks.h, 671  
  
 ume\_block\_src\_create  
     umeblocks.h, 673  
  
 ume\_block\_src\_delete  
     umeblocks.h, 674  
  
 ume\_block\_src\_send\_ex  
     umeblocks.h, 674  
  
 ume\_block\_src\_t\_stct, 289  
  
 ume\_liveness\_receiving\_context\_t  
     lmb.h, 411  
  
 ume\_liveness\_receiving\_context\_t\_stct, 290  
  
 UME\_MALLOC\_RETURN  
     umeblocks.h, 672  
  
 ume\_msg\_clientd

lbm\_src\_send\_ex\_info\_t\_stct, 221  
  
 UME\_SEM\_DESTROY  
     umeblocks.h, 672  
  
 UME\_SEM\_INIT  
     umeblocks.h, 672  
  
 UME\_SEM\_POST  
     umeblocks.h, 672  
  
 UME\_SEM\_TIMEDWAIT  
     umeblocks.h, 673  
  
 UME\_SEM\_WAIT  
     umeblocks.h, 673  
  
 umeblocks.h, 670  
     SLEEP\_MSEC, 671  
  
 UME\_BLOCK\_DEBUG\_PRINT, 671  
  
 UME\_BLOCK\_PRINT\_ERROR, 671  
  
 ume\_block\_src\_create, 673  
  
 ume\_block\_src\_delete, 674  
  
 ume\_block\_src\_send\_ex, 674  
  
 UME\_MALLOC\_RETURN, 672  
  
 UME\_SEM\_DESTROY, 672  
  
 UME\_SEM\_INIT, 672  
  
 UME\_SEM\_POST, 672  
  
 UME\_SEM\_TIMEDWAIT, 673  
  
 UME\_SEM\_WAIT, 673  
  
 umq\_index  
     lbm\_src\_send\_ex\_info\_t\_stct, 222  
  
 umq\_msg\_total\_lifetime  
     lbm\_umq\_msg\_total\_lifetime\_info\_t\_stct, 258  
  
 umq\_total\_lifetime  
     lbm\_src\_send\_ex\_info\_t\_stct, 222  
  
 unblock\_events  
     lbm\_event\_queue\_stats\_t\_stct, 139  
  
 unblock\_events\_tot  
     lbm\_event\_queue\_stats\_t\_stct, 139  
  
 unrecovered\_tmo  
     lbm\_rcv\_transport\_stats\_lbtrm\_t\_stct, 181  
  
     lbm\_rcv\_transport\_stats\_lbtru\_t\_stct, 188  
  
 unrecovered\_txw  
     lbm\_rcv\_transport\_stats\_lbtrm\_t\_stct, 180

lbm\_rcv\_transport\_stats\_lbtru\_t\_-  
stct, [187](#)

username  
[lbm\\_umm\\_info\\_t\\_stct](#), [255](#)

value  
[lbm\\_umq\\_ulb\\_application\\_set\\_attr\\_t\\_stct](#), [265](#)  
[lbm\\_umq\\_ulb\\_receiver\\_type\\_attr\\_t\\_stct](#), [266](#)  
[lbmpdm\\_field\\_value\\_stct\\_t](#), [285](#)

value\_arr  
[lbmpdm\\_field\\_value\\_stct\\_t](#), [286](#)

wrcv\_msgs  
[lbm\\_event\\_queue\\_stats\\_t\\_stct](#), [136](#)

wrcv\_msgs\_svc\_max  
[lbm\\_event\\_queue\\_stats\\_t\\_stct](#), [136](#)

wrcv\_msgs\_svc\_mean  
[lbm\\_event\\_queue\\_stats\\_t\\_stct](#), [136](#)

wrcv\_msgs\_svc\_min  
[lbm\\_event\\_queue\\_stats\\_t\\_stct](#), [136](#)

wrcv\_msgs\_tot  
[lbm\\_event\\_queue\\_stats\\_t\\_stct](#), [136](#)