

# LBM API Reference Manual

6.7.1

Generated by Doxygen 1.4.7

Wed Jul 16 15:57:56 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 . . . . .	131
7.11	lbm_delete_cb_info_t_stct Struct Reference . . . . .	132
7.12	lbm_event_queue_cancel_cb_info_t_stct Struct Reference . . . . .	133
7.13	lbm_event_queue_stats_t_stct Struct Reference . . . . .	134
7.14	lbm_flight_size_inflight_t_stct Struct Reference . . . . .	146
7.15	lbm_hf_sequence_number_t_stct Union Reference . . . . .	147
7.16	lbm_iovec_t_stct Struct Reference . . . . .	148
7.17	lbm_ipv4_address_mask_t_stct Struct Reference . . . . .	149
7.18	lbm_mim_unrecloss_func_t_stct Struct Reference . . . . .	150
7.19	lbm_msg_channel_info_t_stct Struct Reference . . . . .	151

---

7.20	<a href="#">lbm_msg_fragment_info_t_stct Struct Reference</a>	152
7.21	<a href="#">lbm_msg_gateway_info_t_stct Struct Reference</a>	153
7.22	<a href="#">lbm_msg_properties_iter_t_stct Struct Reference</a>	154
7.23	<a href="#">lbm_msg_t_stct Struct Reference</a>	155
7.24	<a href="#">lbm_msg_ume_deregistration_ex_t_stct Struct Reference</a>	160
7.25	<a href="#">lbm_msg_ume_registration_complete_ex_t_stct Struct Reference</a>	162
7.26	<a href="#">lbm_msg_ume_registration_ex_t_stct Struct Reference</a>	163
7.27	<a href="#">lbm_msg_ume_registration_t_stct Struct Reference</a>	165
7.28	<a href="#">lbm_msg_umq_deregistration_complete_ex_t_stct Struct Reference</a>	166
7.29	<a href="#">lbm_msg_umq_index_assigned_ex_t_stct Struct Reference</a>	167
7.30	<a href="#">lbm_msg_umq_index_assignment_eligibility_start_complete_ex_t_stct Struct Reference</a>	168
7.31	<a href="#">lbm_msg_umq_index_assignment_eligibility_stop_complete_ex_t_stct Struct Reference</a>	169
7.32	<a href="#">lbm_msg_umq_index_released_ex_t_stct Struct Reference</a>	170
7.33	<a href="#">lbm_msg_umq_registration_complete_ex_t_stct Struct Reference</a>	171
7.34	<a href="#">lbm_rcv_src_notification_func_t_stct Struct Reference</a>	173
7.35	<a href="#">lbm_rcv_topic_stats_t_stct Struct Reference</a>	174
7.36	<a href="#">lbm_rcv_transport_stats_daemon_t_stct Struct Reference</a>	176
7.37	<a href="#">lbm_rcv_transport_stats_lbtpc_t_stct Struct Reference</a>	177
7.38	<a href="#">lbm_rcv_transport_stats_lbtrdma_t_stct Struct Reference</a>	179
7.39	<a href="#">lbm_rcv_transport_stats_lbtrm_t_stct Struct Reference</a>	181
7.40	<a href="#">lbm_rcv_transport_stats_lbtru_t_stct Struct Reference</a>	188
7.41	<a href="#">lbm_rcv_transport_stats_lbtsmx_t_stct Struct Reference</a>	194
7.42	<a href="#">lbm_rcv_transport_stats_t_stct Struct Reference</a>	196
7.43	<a href="#">lbm_rcv_transport_stats_tcp_t_stct Struct Reference</a>	199
7.44	<a href="#">lbm_rcv_umq_queue_msg_list_info_t Struct Reference</a>	201
7.45	<a href="#">lbm_rcv_umq_queue_msg_retrieve_info_t Struct Reference</a>	202
7.46	<a href="#">lbm_resolver_event_advertisement_t_stct Struct Reference</a>	203
7.47	<a href="#">lbm_resolver_event_func_t_stct Struct Reference</a>	204
7.48	<a href="#">lbm_resolver_event_info_t_stct Struct Reference</a>	205
7.49	<a href="#">lbm_serialized_response_t_stct Struct Reference</a>	206

---

7.50	<a href="#">lbm_src_cost_func_t_stct Struct Reference</a>	207
7.51	<a href="#">lbm_src_event_flight_size_notification_t_stct Struct Reference</a>	208
7.52	<a href="#">lbm_src_event_sequence_number_info_t_stct Struct Reference</a>	209
7.53	<a href="#">lbm_src_event_ume_ack_ex_info_t_stct Struct Reference</a>	211
7.54	<a href="#">lbm_src_event_ume_ack_info_t_stct Struct Reference</a>	213
7.55	<a href="#">lbm_src_event_ume_deregistration_ex_t_stct Struct Reference</a>	214
7.56	<a href="#">lbm_src_event_ume_registration_complete_ex_t_stct Struct Reference</a>	216
7.57	<a href="#">lbm_src_event_ume_registration_ex_t_stct Struct Reference</a>	217
7.58	<a href="#">lbm_src_event_ume_registration_t_stct Struct Reference</a>	219
7.59	<a href="#">lbm_src_event_umq_message_id_info_t_stct Struct Reference</a>	220
7.60	<a href="#">lbm_src_event_umq_registration_complete_ex_t_stct Struct Reference</a>	222
7.61	<a href="#">lbm_src_event_umq_stability_ack_info_ex_t_stct Struct Reference</a>	223
7.62	<a href="#">lbm_src_event_umq_ulb_message_info_ex_t_stct Struct Reference</a>	225
7.63	<a href="#">lbm_src_event_umq_ulb_receiver_info_ex_t_stct Struct Reference</a>	227
7.64	<a href="#">lbm_src_event_wakeup_t_stct Struct Reference</a>	229
7.65	<a href="#">lbm_src_notify_func_t_stct Struct Reference</a>	230
7.66	<a href="#">lbm_src_send_ex_info_t_stct Struct Reference</a>	231
7.67	<a href="#">lbm_src_transport_stats_daemon_t_stct Struct Reference</a>	233
7.68	<a href="#">lbm_src_transport_stats_lbtipc_t_stct Struct Reference</a>	234
7.69	<a href="#">lbm_src_transport_stats_lbtrdma_t_stct Struct Reference</a>	235
7.70	<a href="#">lbm_src_transport_stats_lbtrm_t_stct Struct Reference</a>	236
7.71	<a href="#">lbm_src_transport_stats_lbtru_t_stct Struct Reference</a>	240
7.72	<a href="#">lbm_src_transport_stats_lbtstmx_t_stct Struct Reference</a>	243
7.73	<a href="#">lbm_src_transport_stats_t_stct Struct Reference</a>	244
7.74	<a href="#">lbm_src_transport_stats_tcp_t_stct Struct Reference</a>	247
7.75	<a href="#">lbm_str_hash_func_ex_t_stct Struct Reference</a>	248
7.76	<a href="#">lbm_timeval_t_stct Struct Reference</a>	249
7.77	<a href="#">lbm_transport_source_info_t_stct Struct Reference</a>	250
7.78	<a href="#">lbm_ucast_resolver_entry_t_stct Struct Reference</a>	253
7.79	<a href="#">lbm_ume_ctx_rcv_ctx_notification_func_t_stct Struct Reference</a>	255
7.80	<a href="#">lbm_ume_rcv_recovery_info_ex_func_info_t_stct Struct Reference</a>	256

---

7.81	<a href="#">lbm_ume_rcv_recovery_info_ex_func_t_stct Struct Reference</a>	258
7.82	<a href="#">lbm_ume_rcv_regid_ex_func_info_t_stct Struct Reference</a>	259
7.83	<a href="#">lbm_ume_rcv_regid_ex_func_t_stct Struct Reference</a>	261
7.84	<a href="#">lbm_ume_rcv_regid_func_t_stct Struct Reference</a>	262
7.85	<a href="#">lbm_ume_src_force_reclaim_func_t_stct Struct Reference</a>	263
7.86	<a href="#">lbm_ume_store_entry_t_stct Struct Reference</a>	264
7.87	<a href="#">lbm_ume_store_group_entry_t_stct Struct Reference</a>	266
7.88	<a href="#">lbm_ume_store_name_entry_t_stct Struct Reference</a>	267
7.89	<a href="#">lbm_umm_info_t_stct Struct Reference</a>	268
7.90	<a href="#">lbm_umq_index_info_t_stct Struct Reference</a>	270
7.91	<a href="#">lbm_umq_msg_total_lifetime_info_t_stct Struct Reference</a>	271
7.92	<a href="#">lbm_umq_msgid_t_stct Struct Reference</a>	272
7.93	<a href="#">lbm_umq_queue_entry_t_stct Struct Reference</a>	273
7.94	<a href="#">lbm_umq_queue_msg_status_t Struct Reference</a>	274
7.95	<a href="#">lbm_umq_queue_topic_status_t Struct Reference</a>	276
7.96	<a href="#">lbm_umq_queue_topic_t_stct Struct Reference</a>	277
7.97	<a href="#">lbm_umq_ulb_application_set_attr_t_stct Struct Reference</a>	278
7.98	<a href="#">lbm_umq_ulb_receiver_type_attr_t_stct Struct Reference</a>	279
7.99	<a href="#">lbm_umq_ulb_receiver_type_entry_t_stct Struct Reference</a>	280
7.100	<a href="#">lbm_wildcard_rcv_compare_func_t_stct Struct Reference</a>	281
7.101	<a href="#">lbm_wildcard_rcv_create_func_t_stct Struct Reference</a>	282
7.102	<a href="#">lbm_wildcard_rcv_delete_func_t_stct Struct Reference</a>	283
7.103	<a href="#">lbm_wildcard_rcv_stats_t_stct Struct Reference</a>	284
7.104	<a href="#">lbmmon_attr_block_t_stct Struct Reference</a>	285
7.105	<a href="#">lbmmon_attr_entry_t_stct Struct Reference</a>	286
7.106	<a href="#">lbmmon_ctx_statistics_func_t_stct Struct Reference</a>	287
7.107	<a href="#">lbmmon_evq_statistics_func_t_stct Struct Reference</a>	288
7.108	<a href="#">lbmmon_format_func_t_stct Struct Reference</a>	289
7.109	<a href="#">lbmmon_packet_hdr_t_stct Struct Reference</a>	292
7.110	<a href="#">lbmmon_rcv_statistics_func_t_stct Struct Reference</a>	294
7.111	<a href="#">lbmmon_rcv_topic_statistics_func_t_stct Struct Reference</a>	295

7.112	lbmmon_src_statistics_func_t_stct Struct Reference . . . . .	296
7.113	lbmmon_transport_func_t_stct Struct Reference . . . . .	297
7.114	lbmmon_wildcard_rcv_statistics_func_t_stct Struct Reference . . . . .	299
7.115	lbmpdm_decimal_t Struct Reference . . . . .	300
7.116	lbmpdm_field_info_attr_stct_t Struct Reference . . . . .	301
7.117	lbmpdm_field_value_stct_t Struct Reference . . . . .	302
7.118	lbmpdm_timestamp_t Struct Reference . . . . .	304
7.119	lbmsdm_decimal_t_stct Struct Reference . . . . .	305
7.120	ume_block_src_t_stct Struct Reference . . . . .	306
7.121	ume_liveness_receiving_context_t_stct Struct Reference . . . . .	307
<b>8</b>	<b>LBM API File Documentation</b>	<b>309</b>
8.1	lbm.h File Reference . . . . .	309
8.2	lbmaux.h File Reference . . . . .	557
8.3	lbmht.h File Reference . . . . .	561
8.4	lbmmon.h File Reference . . . . .	567
8.5	lbmpdm.h File Reference . . . . .	605
8.6	lbmsdm.h File Reference . . . . .	645
8.7	umeblocksrc.h File Reference . . . . .	701
<b>9</b>	<b>LBM API Page Documentation</b>	<b>707</b>
9.1	LBMMON Example source code . . . . .	707
9.2	LBMMON LBM transport module . . . . .	708
9.3	Source code for lbmmontrlbm.h . . . . .	709
9.4	Source code for lbmmontrlbm.c . . . . .	712
9.5	LBMMON UDP transport module . . . . .	733
9.6	Source code for lbmmontrudp.h . . . . .	734
9.7	Source code for lbmmontrudp.c . . . . .	737
9.8	LBMMON CSV format module . . . . .	754
9.9	Source code for lbmmonfmtcsv.h . . . . .	755
9.10	Source code for lbmmonfmtcsv.c . . . . .	762
9.11	LBMMON LBMSNMP transport module . . . . .	810

---

9.12 Source code for <code>lbmontrlbmsnmp.h</code> . . . . .	811
9.13 Source code for <code>lbmontrlbmsnmp.c</code> . . . . .	814
9.14 Deprecated List . . . . .	836



## **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:

<a href="#">lbm_apphdr_chain_elem_t_stct</a> (Structure that represents an element in an app header chain) . . . . .	113
<a href="#">lbm_async_operation_func_t</a> (Structure that holds information for asynchronous operation callbacks) . . . . .	115
<a href="#">lbm_async_operation_info_t</a> (Results struct returned via the user-specified asynchronous operation callback from any asynchronous API) . . .	116
<a href="#">lbm_context_event_func_t_stct</a> (Structure that holds the application callback for context-level events) . . . . .	118
<a href="#">lbm_context_event_umq_registration_complete_ex_t_stct</a> (Structure that holds information for contexts after registration is complete to all involved queue instances) . . . . .	119
<a href="#">lbm_context_event_umq_registration_ex_t_stct</a> (Structure that holds queue registration information for the UMQ context in an extended form) .	121
<a href="#">lbm_context_rcv_immediate_msgs_func_t_stct</a> (Structure that holds the application callback for receiving topic-less immediate mode messages) . . . . .	123
<a href="#">lbm_context_src_event_func_t_stct</a> (Structure that holds the application callback for context-level source events) . . . . .	124
<a href="#">lbm_context_stats_t_stct</a> (Structure that holds statistics for a context) . . . .	125
<a href="#">lbm_ctx_umq_queue_topic_list_info_t</a> (Struct containing an array of queue topics retrieved via <code>lbm_umq_queue_topic_list</code> ) . . . . .	131
<a href="#">lbm_delete_cb_info_t_stct</a> (Structure passed to the <code>lbm_hypertopic_rcv_delete()</code> function so that a deletion callback may be called) . . . . .	132
<a href="#">lbm_event_queue_cancel_cb_info_t_stct</a> (Structure passed to cancel/delete functions so that a cancel callback may be called) . . . . .	133

<a href="#">lbm_event_queue_stats_t_stct</a> (Structure that holds statistics for an event queue) . . . . .	134
<a href="#">lbm_flight_size_inflight_t_stct</a> (Structure that holds information for source total inflight messages and bytes) . . . . .	146
<a href="#">lbm_hf_sequence_number_t_stct</a> (Structure to hold a hot failover sequence number) . . . . .	147
<a href="#">lbm_iovec_t_stct</a> (Structure, struct iovec compatible, that holds information about buffers used for vectored sends) . . . . .	148
<a href="#">lbm_ipv4_address_mask_t_stct</a> (Structure that holds an IPv4 address and a CIDR style netmask) . . . . .	149
<a href="#">lbm_mim_unrecloss_func_t_stct</a> (Structure that holds the application callback for multicast immediate message unrecoverable loss notification) . . . . .	150
<a href="#">lbm_msg_channel_info_t_stct</a> (Structure that represents UMS Spectrum channel information) . . . . .	151
<a href="#">lbm_msg_fragment_info_t_stct</a> (Structure that holds fragment information for UM messages when appropriate) . . . . .	152
<a href="#">lbm_msg_gateway_info_t_stct</a> (Structure that holds originating information for UM messages which arrived via a gateway) . . . . .	153
<a href="#">lbm_msg_properties_iter_t_stct</a> (A struct used for iterating over properties pointed to by an <code>lbm_msg_properties_t</code> ) . . . . .	154
<a href="#">lbm_msg_t_stct</a> (Structure that stores information about a received message) . . . . .	155
<a href="#">lbm_msg_ume_deregistration_ex_t_stct</a> (Structure that holds store deregistration information for the UM receiver in an extended form) . . . . .	160
<a href="#">lbm_msg_ume_registration_complete_ex_t_stct</a> (Structure that holds information for receivers after registration is complete to all involved stores) . . . . .	162
<a href="#">lbm_msg_ume_registration_ex_t_stct</a> (Structure that holds store registration information for the UM receiver in an extended form) . . . . .	163
<a href="#">lbm_msg_ume_registration_t_stct</a> (Structure that holds store registration information for the UMP receiver) . . . . .	165
<a href="#">lbm_msg_umq_deregistration_complete_ex_t_stct</a> (Structure that holds information for receivers after they de-register from a queue) . . . . .	166
<a href="#">lbm_msg_umq_index_assigned_ex_t_stct</a> (Structure that holds beginning-of-index information for receivers) . . . . .	167
<a href="#">lbm_msg_umq_index_assignment_eligibility_start_complete_ex_t_stct</a> (Structure that holds index assignment information for receivers) . . . . .	168
<a href="#">lbm_msg_umq_index_assignment_eligibility_stop_complete_ex_t_stct</a> (Structure that holds index assignment information for receivers) . . . . .	169
<a href="#">lbm_msg_umq_index_released_ex_t_stct</a> (Structure that holds end-of-index information for receivers) . . . . .	170
<a href="#">lbm_msg_umq_registration_complete_ex_t_stct</a> (Structure that holds information for receivers after registration is complete to all involved queue instances) . . . . .	171
<a href="#">lbm_rcv_src_notification_func_t_stct</a> (Structure that holds the application callback for source status notifications for receivers) . . . . .	173

<a href="#">lbm_rcv_topic_stats_t_stct</a> (Structure that holds statistics for a receiver topic )	174
<a href="#">lbm_rcv_transport_stats_daemon_t_stct</a> (Structure that holds statistics for receiver daemon mode transport (deprecated) ) . . . . .	176
<a href="#">lbm_rcv_transport_stats_lbtpc_t_stct</a> (Structure that holds datagram statistics for receiver LBT-IPC transports ) . . . . .	177
<a href="#">lbm_rcv_transport_stats_lbtrdma_t_stct</a> (Structure that holds datagram statistics for receiver LBT-RDMA transports ) . . . . .	179
<a href="#">lbm_rcv_transport_stats_lbtrm_t_stct</a> (Structure that holds datagram statistics for receiver LBT-RM transports ) . . . . .	181
<a href="#">lbm_rcv_transport_stats_lbtru_t_stct</a> (Structure that holds datagram statistics for receiver LBT-RU transports ) . . . . .	188
<a href="#">lbm_rcv_transport_stats_lbtsmx_t_stct</a> (Structure that holds datagram statistics for receiver LBT-SMX transports ) . . . . .	194
<a href="#">lbm_rcv_transport_stats_t_stct</a> (Structure that holds statistics for receiver transports ) . . . . .	196
<a href="#">lbm_rcv_transport_stats_tcp_t_stct</a> (Structure that holds datagram statistics for receiver TCP transports ) . . . . .	199
<a href="#">lbm_rcv_umq_queue_msg_list_info_t</a> (Struct containing an array of UMQ messages listed via <code>lbm_rcv_umq_queue_msg_list</code> ) . . . . .	201
<a href="#">lbm_rcv_umq_queue_msg_retrieve_info_t</a> (Struct containing an array of UMQ messages retrieved via <code>lbm_rcv_umq_queue_msg_retrieve</code> ) .	202
<a href="#">lbm_resolver_event_advertisement_t_stct</a> (Advertisement event structure (for internal use only) ) . . . . .	203
<a href="#">lbm_resolver_event_func_t_stct</a> (Resolver event function (for internal use only) ) . . . . .	204
<a href="#">lbm_resolver_event_info_t_stct</a> (Resolver event structure (for internal use only) ) . . . . .	205
<a href="#">lbm_serialized_response_t_stct</a> (Structure that holds a serialized UM response object ) . . . . .	206
<a href="#">lbm_src_cost_func_t_stct</a> (Structure that holds the "source_cost_evaluation_function" context attribute ) . . . . .	207
<a href="#">lbm_src_event_flight_size_notification_t_stct</a> (Structure that holds flight size notification event data ) . . . . .	208
<a href="#">lbm_src_event_sequence_number_info_t_stct</a> (Structure that holds sequence number information for a message sent by a source ) . . . . .	209
<a href="#">lbm_src_event_ume_ack_ex_info_t_stct</a> (Structure that holds ACK information for a given message in an extended form ) . . . . .	211
<a href="#">lbm_src_event_ume_ack_info_t_stct</a> (Structure that holds ACK information for a given message ) . . . . .	213
<a href="#">lbm_src_event_ume_deregistration_ex_t_stct</a> (Structure that holds store deregistration information for the UMP source in an extended form )	214
<a href="#">lbm_src_event_ume_registration_complete_ex_t_stct</a> (Structure that holds information for sources after registration is complete to all involved stores ) . . . . .	216
<a href="#">lbm_src_event_ume_registration_ex_t_stct</a> (Structure that holds store registration information for the UMP source in an extended form ) . . . .	217

<a href="#">lbm_src_event_ume_registration_t_stct</a> (Structure that holds store registration information for the UMP source ) . . . . .	219
<a href="#">lbm_src_event_umq_message_id_info_t_stct</a> (Structure that holds Message ID information for a message sent by a sending UMQ application ) . . . . .	220
<a href="#">lbm_src_event_umq_registration_complete_ex_t_stct</a> (Structure that holds information for sources after registration is complete to all involved queue instances ) . . . . .	222
<a href="#">lbm_src_event_umq_stability_ack_info_ex_t_stct</a> (Structure that holds UMQ ACK information for a given message in an extended form ) . . . . .	223
<a href="#">lbm_src_event_umq_ulb_message_info_ex_t_stct</a> (Structure that holds UMQ ULB message information in an extended form ) . . . . .	225
<a href="#">lbm_src_event_umq_ulb_receiver_info_ex_t_stct</a> (Structure that holds UMQ ULB receiver information in an extended form ) . . . . .	227
<a href="#">lbm_src_event_wakeup_t_stct</a> (Structure that holds source wakeup event data ) . . . . .	229
<a href="#">lbm_src_notify_func_t_stct</a> (Structure that holds the callback for source notifications ) . . . . .	230
<a href="#">lbm_src_send_ex_info_t_stct</a> (Structure that holds information for the extended send calls A structure used with UM sources that utilize the extended send calls to pass options ) . . . . .	231
<a href="#">lbm_src_transport_stats_daemon_t_stct</a> (Structure that holds statistics for source daemon mode transport (deprecated) ) . . . . .	233
<a href="#">lbm_src_transport_stats_lbtipc_t_stct</a> (Structure that holds datagram statistics for source LBT-IPC transports ) . . . . .	234
<a href="#">lbm_src_transport_stats_lbtrdma_t_stct</a> (Structure that holds datagram statistics for source LBT-RDMA transports ) . . . . .	235
<a href="#">lbm_src_transport_stats_lbrm_t_stct</a> (Structure that holds datagram statistics for source LBT-RM transports ) . . . . .	236
<a href="#">lbm_src_transport_stats_lbtru_t_stct</a> (Structure that holds datagram statistics for source LBT-RU transports ) . . . . .	240
<a href="#">lbm_src_transport_stats_lbtsmx_t_stct</a> (Structure that holds datagram statistics for source LBT-SMX transports ) . . . . .	243
<a href="#">lbm_src_transport_stats_t_stct</a> (Structure that holds statistics for source transports ) . . . . .	244
<a href="#">lbm_src_transport_stats_tcp_t_stct</a> (Structure that holds datagram statistics for source TCP transports ) . . . . .	247
<a href="#">lbm_str_hash_func_ex_t_stct</a> (Structure that holds the hash function callback information ) . . . . .	248
<a href="#">lbm_timeval_t_stct</a> (Structure that holds seconds and microseconds since midnight, Jan 1, 1970 UTC ) . . . . .	249
<a href="#">lbm_transport_source_info_t_stct</a> (Structure that holds formatted and parsed transport source strings ) . . . . .	250
<a href="#">lbm_ucast_resolver_entry_t_stct</a> (Structure that holds information for a unicast resolver daemon for configuration purposes ) . . . . .	253

<a href="#">lbm_ume_ctx_rcv_ctx_notification_func_t_stct</a> (Structure that holds the application callback for receiving context status notifications for source context ) . . . . .	255
<a href="#">lbm_ume_rcv_recovery_info_ex_func_info_t_stct</a> (Structure that holds information for UMP receiver recovery sequence number info application callbacks ) . . . . .	256
<a href="#">lbm_ume_rcv_recovery_info_ex_func_t_stct</a> (Structure that holds the application callback for recovery sequence number information, extended form ) . . . . .	258
<a href="#">lbm_ume_rcv_regid_ex_func_info_t_stct</a> (Structure that holds information for UMP receiver registration ID application callbacks ) . . . . .	259
<a href="#">lbm_ume_rcv_regid_ex_func_t_stct</a> (Structure that holds the application callback for registration ID setting, extended form ) . . . . .	261
<a href="#">lbm_ume_rcv_regid_func_t_stct</a> (Structure that holds the application callback for registration ID setting ) . . . . .	262
<a href="#">lbm_ume_src_force_reclaim_func_t_stct</a> (Structure that holds the application callback for forced reclamation notifications ) . . . . .	263
<a href="#">lbm_ume_store_entry_t_stct</a> (Structure that holds information for a UMP store for configuration purposes ) . . . . .	264
<a href="#">lbm_ume_store_group_entry_t_stct</a> (Structure that holds information for a UMP store group for configuration purposes ) . . . . .	266
<a href="#">lbm_ume_store_name_entry_t_stct</a> (Structure that holds information for a UMP store by name for configuration purposes ) . . . . .	267
<a href="#">lbm_umm_info_t_stct</a> (Structure for specifying UMM daemon connection options ) . . . . .	268
<a href="#">lbm_umq_index_info_t_stct</a> (Structure that holds information used for sending and receiving messages with UMQ indices ) . . . . .	270
<a href="#">lbm_umq_msg_total_lifetime_info_t_stct</a> (Structure that holds UMQ message total lifetime information ) . . . . .	271
<a href="#">lbm_umq_msgid_t_stct</a> (Structure that holds information for UMQ messages that allows the message to be identified uniquely ) . . . . .	272
<a href="#">lbm_umq_queue_entry_t_stct</a> (Structure that holds information for a UMQ queue registration ID for configuration purposes ) . . . . .	273
<a href="#">lbm_umq_queue_msg_status_t</a> (Struct containing extended asynchronous operation status information about a single UMQ message ) . . . . .	274
<a href="#">lbm_umq_queue_topic_status_t</a> (Struct containing extended asynchronous operation status information about a single UMQ topic ) . . . . .	276
<a href="#">lbm_umq_queue_topic_t_stct</a> (Structure that holds queue topic information and can be used as a handle to a queue topic ) . . . . .	277
<a href="#">lbm_umq_ulb_application_set_attr_t_stct</a> (Structure that holds information for a UMQ ULB sources application set attributes ) . . . . .	278
<a href="#">lbm_umq_ulb_receiver_type_attr_t_stct</a> (Structure that holds information for a UMQ ULB sources receiver type attributes ) . . . . .	279
<a href="#">lbm_umq_ulb_receiver_type_entry_t_stct</a> (Structure that holds information for a UMQ ULB sources receiver type associations with application sets ) . . . . .	280

<a href="#">lbm_wildcard_rcv_compare_func_t_stct</a> (Structure that holds the application callback pattern type information for wildcard receivers) . . . . .	281
<a href="#">lbm_wildcard_rcv_create_func_t_stct</a> (Structure that holds the receiver creation callback information for wildcard receivers) . . . . .	282
<a href="#">lbm_wildcard_rcv_delete_func_t_stct</a> (Structure that holds the receiver deletion callback information for wildcard receivers) . . . . .	283
<a href="#">lbm_wildcard_rcv_stats_t_stct</a> (Structure that holds statistics for a wildcard receiver) . . . . .	284
<a href="#">lbmmon_attr_block_t_stct</a> (Statistics attribute block layout. Associated with each statistics message is a set of optional attributes. Any attributes present will immediately follow the packet header) . . . . .	285
<a href="#">lbmmon_attr_entry_t_stct</a> (Statistics attribute entry layout. Each attribute entry within the attributes block consists of an entry header, followed immediately by the attribute data) . . . . .	286
<a href="#">lbmmon_ctx_statistics_func_t_stct</a> (A structure that holds the callback information for context statistics) . . . . .	287
<a href="#">lbmmon_evq_statistics_func_t_stct</a> (A structure that holds the callback information for event queue statistics) . . . . .	288
<a href="#">lbmmon_format_func_t_stct</a> (Format module function pointer container) . . . . .	289
<a href="#">lbmmon_packet_hdr_t_stct</a> (Statistics packet header layout) . . . . .	292
<a href="#">lbmmon_rcv_statistics_func_t_stct</a> (A structure that holds the callback information for receiver statistics) . . . . .	294
<a href="#">lbmmon_rcv_topic_statistics_func_t_stct</a> (For internal use only. A structure that holds the callback information for receiver topic statistics) . . . . .	295
<a href="#">lbmmon_src_statistics_func_t_stct</a> (A structure that holds the callback information for source statistics) . . . . .	296
<a href="#">lbmmon_transport_func_t_stct</a> (Transport module function pointer container) . . . . .	297
<a href="#">lbmmon_wildcard_rcv_statistics_func_t_stct</a> (A structure that holds the callback information for wildcard receiver statistics) . . . . .	299
<a href="#">lbmpdm_decimal_t</a> (Structure to hold a scaled decimal number. A scaled decimal number consists of a mantissa <i>m</i> and an exponent <i>exp</i> . It represents the value $m \cdot 10^{exp}$ ) . . . . .	300
<a href="#">lbmpdm_field_info_attr_stct_t</a> (Attribute struct to be passed along with the name when adding field info to a definition) . . . . .	301
<a href="#">lbmpdm_field_value_stct_t</a> (Field value struct that can be populated with a field value when passed to the <code>lbmpdm_msg_get_field_value_stct</code> function) . . . . .	302
<a href="#">lbmpdm_timestamp_t</a> (Structure to hold a timestamp value) . . . . .	304
<a href="#">lbmsdm_decimal_t_stct</a> (Structure to hold a scaled decimal number. A scaled decimal number consists of a mantissa <i>m</i> and an exponent <i>exp</i> . It represents the value $m \cdot 10^{exp}$ ) . . . . .	305
<a href="#">ume_block_src_t_stct</a> (Structure used to designate an UME Block source) . . . . .	306
<a href="#">ume_liveness_receiving_context_t_stct</a> (Structure that holds the information about a receiving context) . . . . .	307

# Chapter 4

# LBM API File Index

## 4.1 LBM API File List

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

- [lbm.h](#) (Ultra Messaging (UM) API) . . . . . 309
- [lbmaux.h](#) (Ultra Messaging (UM) Auxiliary Functions API) . . . . . 557
- [lbmht.h](#) (Ultra Messaging (UM) HyperTopic API) . . . . . 561
- [lbmmon.h](#) (Ultra Messaging (UM) Monitoring API) . . . . . 567
- [lbmpdm.h](#) (Ultra Messaging (UM) Pre-Defined Message (PDM) API) . . . . 605
- [lbmsdm.h](#) (Ultra Messaging (UM) Self-Describing Message (SDM) API) . . . 645
- [umeblocksrc.h](#) (UME Blocking API) . . . . . 701



# 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 . . . . . [707](#)
- Deprecated List . . . . . [836](#)



## Chapter 6

# LBM API Module Documentation

### 6.1 Add a field to a message

#### Functions

- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_boolean](#) (lbmsdm\_msg\_t \*Message, const char \*Name, uint8\_t Value)  
*Add a field to a message.*
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_int8](#) (lbmsdm\_msg\_t \*Message, const char \*Name, int8\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_uint8](#) (lbmsdm\_msg\_t \*Message, const char \*Name, uint8\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_int16](#) (lbmsdm\_msg\_t \*Message, const char \*Name, int16\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_uint16](#) (lbmsdm\_msg\_t \*Message, const char \*Name, uint16\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_int32](#) (lbmsdm\_msg\_t \*Message, const char \*Name, int32\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_uint32](#) (lbmsdm\_msg\_t \*Message, const char \*Name, uint32\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_int64](#) (lbmsdm\_msg\_t \*Message, const char \*Name, int64\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_uint64](#) (lbmsdm\_msg\_t \*Message, const char \*Name, uint64\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_float](#) (lbmsdm\_msg\_t \*Message, const char \*Name, float Value)

- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_double](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, double Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_decimal](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, const [lbmsdm\\_decimal\\_t](#) \*Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_timestamp](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, const struct timeval \*Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_message](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, const [lbmsdm\\_msg\\_t](#) \*Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_string](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, const char \*Value)
- LBMSDMEpDLL 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.*

- LBMSDMEpDLL 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 LBMSDMEpDLL 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** `LBMSDMEExpDLL 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** `LBMSDMEExpDLL 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** `LBMSDMEExpDLL 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** `LBMSDMEExpDLL 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** `LBMSDMEExpDLL 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** `LBMSDMEExpDLL 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** `LBMSDMEExpDLL 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

- `LBMSDMEpDLL int lbmsdm_msg_add_boolean_array (lbmsdm_msg_t *Message, const char *Name)`  
*Add an array field to a message.*
- `LBMSDMEpDLL int lbmsdm_msg_add_int8_array (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_add_uint8_array (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_add_int16_array (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_add_uint16_array (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_add_int32_array (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_add_uint32_array (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_add_int64_array (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_add_uint64_array (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_add_float_array (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_add_double_array (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_add_decimal_array (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_add_timestamp_array (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_add_message_array (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_add_string_array (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_add_unicode_array (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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

- `LBMSDMEpDLL 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.*

- `LBMSDMEpDLL int lbmsdm_msg_add_int8_elem_idx (lbmsdm_msg_t *Message, size_t Index, int8_t Value)`
- `LBMSDMEpDLL int lbmsdm_msg_add_uint8_elem_idx (lbmsdm_msg_t *Message, size_t Index, uint8_t Value)`
- `LBMSDMEpDLL int lbmsdm_msg_add_int16_elem_idx (lbmsdm_msg_t *Message, size_t Index, int16_t Value)`
- `LBMSDMEpDLL int lbmsdm_msg_add_uint16_elem_idx (lbmsdm_msg_t *Message, size_t Index, uint16_t Value)`
- `LBMSDMEpDLL int lbmsdm_msg_add_int32_elem_idx (lbmsdm_msg_t *Message, size_t Index, int32_t Value)`
- `LBMSDMEpDLL int lbmsdm_msg_add_uint32_elem_idx (lbmsdm_msg_t *Message, size_t Index, uint32_t Value)`
- `LBMSDMEpDLL int lbmsdm_msg_add_int64_elem_idx (lbmsdm_msg_t *Message, size_t Index, int64_t Value)`
- `LBMSDMEpDLL int lbmsdm_msg_add_uint64_elem_idx (lbmsdm_msg_t *Message, size_t Index, uint64_t Value)`
- `LBMSDMEpDLL int lbmsdm_msg_add_float_elem_idx (lbmsdm_msg_t *Message, size_t Index, float Value)`
- `LBMSDMEpDLL int lbmsdm_msg_add_double_elem_idx (lbmsdm_msg_t *Message, size_t Index, double Value)`
- `LBMSDMEpDLL int lbmsdm_msg_add_decimal_elem_idx (lbmsdm_msg_t *Message, size_t Index, const lbmsdm_decimal_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_add_timestamp_elem_idx (lbmsdm_msg_t *Message, size_t Index, const struct timeval *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_add_message_elem_idx (lbmsdm_msg_t *Message, size_t Index, const lbmsdm_msg_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_add_string_elem_idx (lbmsdm_msg_t *Message, size_t Index, const char *Value)`
- `LBMSDMEpDLL 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.*

- `LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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

- `LBMSDMEExpDLL 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.*

- `LBMSDMEExpDLL int lbmsdm_msg_add_int8_elem_name (lbmsdm_msg_t *Message, const char *Name, int8_t Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_add_uint8_elem_name (lbmsdm_msg_t *Message, const char *Name, uint8_t Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_add_int16_elem_name (lbmsdm_msg_t *Message, const char *Name, int16_t Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_add_uint16_elem_name (lbmsdm_msg_t *Message, const char *Name, uint16_t Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_add_int32_elem_name (lbmsdm_msg_t *Message, const char *Name, int32_t Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_add_uint32_elem_name (lbmsdm_msg_t *Message, const char *Name, uint32_t Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_add_int64_elem_name (lbmsdm_msg_t *Message, const char *Name, int64_t Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_add_uint64_elem_name (lbmsdm_msg_t *Message, const char *Name, uint64_t Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_add_float_elem_name (lbmsdm_msg_t *Message, const char *Name, float Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_add_double_elem_name (lbmsdm_msg_t *Message, const char *Name, double Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_add_decimal_elem_name (lbmsdm_msg_t *Message, const char *Name, const lbmsdm_decimal_t *Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_add_timestamp_elem_name (lbmsdm_msg_t *Message, const char *Name, const struct timeval *Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_add_message_elem_name (lbmsdm_msg_t *Message, const char *Name, const lbmsdm_msg_t *Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_add_string_elem_name (lbmsdm_msg_t *Message, const char *Name, const char *Value)`
- `LBMSDMEExpDLL 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.*

- `LBMSDMEExpDLL 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

- `LBMSDMEpDLL 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.*

- `LBMSDMEpDLL int lbmsdm_iter_add_int8_elem (lbmsdm_iter_t *Iterator, int8_t Value)`
- `LBMSDMEpDLL int lbmsdm_iter_add_uint8_elem (lbmsdm_iter_t *Iterator, uint8_t Value)`
- `LBMSDMEpDLL int lbmsdm_iter_add_int16_elem (lbmsdm_iter_t *Iterator, int16_t Value)`
- `LBMSDMEpDLL int lbmsdm_iter_add_uint16_elem (lbmsdm_iter_t *Iterator, uint16_t Value)`
- `LBMSDMEpDLL int lbmsdm_iter_add_int32_elem (lbmsdm_iter_t *Iterator, int32_t Value)`
- `LBMSDMEpDLL int lbmsdm_iter_add_uint32_elem (lbmsdm_iter_t *Iterator, uint32_t Value)`
- `LBMSDMEpDLL int lbmsdm_iter_add_int64_elem (lbmsdm_iter_t *Iterator, int64_t Value)`
- `LBMSDMEpDLL int lbmsdm_iter_add_uint64_elem (lbmsdm_iter_t *Iterator, uint64_t Value)`
- `LBMSDMEpDLL int lbmsdm_iter_add_float_elem (lbmsdm_iter_t *Iterator, float Value)`
- `LBMSDMEpDLL int lbmsdm_iter_add_double_elem (lbmsdm_iter_t *Iterator, double Value)`
- `LBMSDMEpDLL int lbmsdm_iter_add_decimal_elem (lbmsdm_iter_t *Iterator, const lbmsdm_decimal_t *Value)`
- `LBMSDMEpDLL int lbmsdm_iter_add_timestamp_elem (lbmsdm_iter_t *Iterator, const struct timeval *Value)`
- `LBMSDMEpDLL int lbmsdm_iter_add_message_elem (lbmsdm_iter_t *Iterator, const lbmsdm_msg_t *Value)`
- `LBMSDMEpDLL int lbmsdm_iter_add_string_elem (lbmsdm_iter_t *Iterator, const char *Value)`
- `LBMSDMEpDLL 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.*

- `LBMSDMEpDLL 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

- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_int8\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, int8\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_uint8\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, uint8\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_int16\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, int16\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_uint16\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, uint16\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_int32\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, int32\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_uint32\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, uint32\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_int64\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, int64\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_uint64\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, uint64\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_float\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, float \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_double\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, double \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_decimal\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, lbmsdm\_decimal\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_timestamp\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, struct timeval \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_message\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, lbmsdm\_msg\_t \*\*Value)
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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

- `LBMSDMEpDLL 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.*

- `LBMSDMEpDLL int lbmsdm_msg_get_int8_name (lbmsdm_msg_t *Message, const char *Name, int8_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_uint8_name (lbmsdm_msg_t *Message, const char *Name, uint8_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_int16_name (lbmsdm_msg_t *Message, const char *Name, int16_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_uint16_name (lbmsdm_msg_t *Message, const char *Name, uint16_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_int32_name (lbmsdm_msg_t *Message, const char *Name, int32_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_uint32_name (lbmsdm_msg_t *Message, const char *Name, uint32_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_int64_name (lbmsdm_msg_t *Message, const char *Name, int64_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_uint64_name (lbmsdm_msg_t *Message, const char *Name, uint64_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_float_name (lbmsdm_msg_t *Message, const char *Name, float *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_double_name (lbmsdm_msg_t *Message, const char *Name, double *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_decimal_name (lbmsdm_msg_t *Message, const char *Name, lbmsdm_decimal_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_timestamp_name (lbmsdm_msg_t *Message, const char *Name, struct timeval *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_message_name (lbmsdm_msg_t *Message, const char *Name, lbmsdm_msg_t **Value)`
- `LBMSDMEpDLL 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.*

- `LBMSDMEpDLL 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.*

- `LBMSDMEpDLL 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** `LBMSDMEExpDLL 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** `LBMSDMEExpDLL 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** `LBMSDMEExpDLL 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** `LBMSDMEExpDLL 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** `LBMSDMEExpDLL 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** `LBMSDMEExpDLL 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** `LBMSDMEExpDLL 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** `LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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

- `LBMSDMEpDLL 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.*

- `LBMSDMEpDLL int lbmsdm_msg_get_int8_elem_idx (lbmsdm_msg_t *Message, size_t Index, size_t Element, int8_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_uint8_elem_idx (lbmsdm_msg_t *Message, size_t Index, size_t Element, uint8_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_int16_elem_idx (lbmsdm_msg_t *Message, size_t Index, size_t Element, int16_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_uint16_elem_idx (lbmsdm_msg_t *Message, size_t Index, size_t Element, uint16_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_int32_elem_idx (lbmsdm_msg_t *Message, size_t Index, size_t Element, int32_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_uint32_elem_idx (lbmsdm_msg_t *Message, size_t Index, size_t Element, uint32_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_int64_elem_idx (lbmsdm_msg_t *Message, size_t Index, size_t Element, int64_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_uint64_elem_idx (lbmsdm_msg_t *Message, size_t Index, size_t Element, uint64_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_float_elem_idx (lbmsdm_msg_t *Message, size_t Index, size_t Element, float *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_double_elem_idx (lbmsdm_msg_t *Message, size_t Index, size_t Element, double *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_decimal_elem_idx (lbmsdm_msg_t *Message, size_t Index, size_t Element, lbmsdm_decimal_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_timestamp_elem_idx (lbmsdm_msg_t *Message, size_t Index, size_t Element, struct timeval *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_get_message_elem_idx (lbmsdm_msg_t *Message, size_t Index, size_t Element, lbmsdm_msg_t **Value)`
- `LBMSDMEpDLL 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.*

- `LBMSDMEpDLL 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.*

- `LBMSDMEpDLL 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

- `LBMSDMEExpDLL 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.*

- `LBMSDMEExpDLL int lbmsdm_msg_get_int8_elem_name (lbmsdm_msg_t *Message, const char *Name, size_t Element, int8_t *Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_get_uint8_elem_name (lbmsdm_msg_t *Message, const char *Name, size_t Element, uint8_t *Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_get_int16_elem_name (lbmsdm_msg_t *Message, const char *Name, size_t Element, int16_t *Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_get_uint16_elem_name (lbmsdm_msg_t *Message, const char *Name, size_t Element, uint16_t *Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_get_int32_elem_name (lbmsdm_msg_t *Message, const char *Name, size_t Element, int32_t *Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_get_uint32_elem_name (lbmsdm_msg_t *Message, const char *Name, size_t Element, uint32_t *Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_get_int64_elem_name (lbmsdm_msg_t *Message, const char *Name, size_t Element, int64_t *Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_get_uint64_elem_name (lbmsdm_msg_t *Message, const char *Name, size_t Element, uint64_t *Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_get_float_elem_name (lbmsdm_msg_t *Message, const char *Name, size_t Element, float *Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_get_double_elem_name (lbmsdm_msg_t *Message, const char *Name, size_t Element, double *Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_get_decimal_elem_name (lbmsdm_msg_t *Message, const char *Name, size_t Element, lbmsdm_decimal_t *Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_get_timestamp_elem_name (lbmsdm_msg_t *Message, const char *Name, size_t Element, struct timeval *Value)`
- `LBMSDMEExpDLL int lbmsdm_msg_get_message_elem_name (lbmsdm_msg_t *Message, const char *Name, size_t Element, lbmsdm_msg_t **Value)`
- `LBMSDMEExpDLL 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.*

- `LBMSDMEExpDLL 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.*

- `LBMSDMEExpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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.*

- LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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

- `LBMSDMEpDLL 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.*

- `LBMSDMEpDLL int lbmsdm_msg_set_int8_idx (lbmsdm_msg_t *Message, size_t Index, int8_t Value)`
- `LBMSDMEpDLL int lbmsdm_msg_set_uint8_idx (lbmsdm_msg_t *Message, size_t Index, uint8_t Value)`
- `LBMSDMEpDLL int lbmsdm_msg_set_int16_idx (lbmsdm_msg_t *Message, size_t Index, int16_t Value)`
- `LBMSDMEpDLL int lbmsdm_msg_set_uint16_idx (lbmsdm_msg_t *Message, size_t Index, uint16_t Value)`
- `LBMSDMEpDLL int lbmsdm_msg_set_int32_idx (lbmsdm_msg_t *Message, size_t Index, int32_t Value)`
- `LBMSDMEpDLL int lbmsdm_msg_set_uint32_idx (lbmsdm_msg_t *Message, size_t Index, uint32_t Value)`
- `LBMSDMEpDLL int lbmsdm_msg_set_int64_idx (lbmsdm_msg_t *Message, size_t Index, int64_t Value)`
- `LBMSDMEpDLL int lbmsdm_msg_set_uint64_idx (lbmsdm_msg_t *Message, size_t Index, uint64_t Value)`
- `LBMSDMEpDLL int lbmsdm_msg_set_float_idx (lbmsdm_msg_t *Message, size_t Index, float Value)`
- `LBMSDMEpDLL int lbmsdm_msg_set_double_idx (lbmsdm_msg_t *Message, size_t Index, double Value)`
- `LBMSDMEpDLL int lbmsdm_msg_set_decimal_idx (lbmsdm_msg_t *Message, size_t Index, const lbmsdm_decimal_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_set_timestamp_idx (lbmsdm_msg_t *Message, size_t Index, const struct timeval *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_set_message_idx (lbmsdm_msg_t *Message, size_t Index, const lbmsdm_msg_t *Value)`
- `LBMSDMEpDLL int lbmsdm_msg_set_string_idx (lbmsdm_msg_t *Message, size_t Index, const char *Value)`
- `LBMSDMEpDLL 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.*

- `LBMSDMEpDLL 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** `LBMSDMExpDLL 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

- `LBMSDMEExpDLL` `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.*
- `LBMSDMEExpDLL` `int` `lbmsdm_msg_set_int8_name` (`lbmsdm_msg_t`  
`*Message`, `const char *Name`, `int8_t Value`)
- `LBMSDMEExpDLL` `int` `lbmsdm_msg_set_uint8_name` (`lbmsdm_msg_t`  
`*Message`, `const char *Name`, `uint8_t Value`)
- `LBMSDMEExpDLL` `int` `lbmsdm_msg_set_int16_name` (`lbmsdm_msg_t`  
`*Message`, `const char *Name`, `int16_t Value`)
- `LBMSDMEExpDLL` `int` `lbmsdm_msg_set_uint16_name` (`lbmsdm_msg_t`  
`*Message`, `const char *Name`, `uint16_t Value`)
- `LBMSDMEExpDLL` `int` `lbmsdm_msg_set_int32_name` (`lbmsdm_msg_t`  
`*Message`, `const char *Name`, `int32_t Value`)
- `LBMSDMEExpDLL` `int` `lbmsdm_msg_set_uint32_name` (`lbmsdm_msg_t`  
`*Message`, `const char *Name`, `uint32_t Value`)
- `LBMSDMEExpDLL` `int` `lbmsdm_msg_set_int64_name` (`lbmsdm_msg_t`  
`*Message`, `const char *Name`, `int64_t Value`)
- `LBMSDMEExpDLL` `int` `lbmsdm_msg_set_uint64_name` (`lbmsdm_msg_t`  
`*Message`, `const char *Name`, `uint64_t Value`)
- `LBMSDMEExpDLL` `int` `lbmsdm_msg_set_float_name` (`lbmsdm_msg_t`  
`*Message`, `const char *Name`, `float Value`)
- `LBMSDMEExpDLL` `int` `lbmsdm_msg_set_double_name` (`lbmsdm_msg_t`  
`*Message`, `const char *Name`, `double Value`)
- `LBMSDMEExpDLL` `int` `lbmsdm_msg_set_decimal_name` (`lbmsdm_msg_t`  
`*Message`, `const char *Name`, `const lbmsdm_decimal_t *Value`)
- `LBMSDMEExpDLL` `int` `lbmsdm_msg_set_timestamp_name` (`lbmsdm_msg_t`  
`*Message`, `const char *Name`, `const struct timeval *Value`)
- `LBMSDMEExpDLL` `int` `lbmsdm_msg_set_message_name` (`lbmsdm_msg_t`  
`*Message`, `const char *Name`, `const lbmsdm_msg_t *Value`)
- `LBMSDMEExpDLL` `int` `lbmsdm_msg_set_string_name` (`lbmsdm_msg_t`  
`*Message`, `const char *Name`, `const char *Value`)
- `LBMSDMEExpDLL` `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.*
- `LBMSDMEExpDLL` `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 `LBMSDMEExpDLL 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 `LBMSDMEExpDLL 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 `LBMSDMEExpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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

- LBMSDMEpDLL int `lbmsdm_iter_set_boolean` (`lbmsdm_iter_t` \*Iterator, uint8\_t Value)

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

- LBMSDMEpDLL int `lbmsdm_iter_set_int8` (`lbmsdm_iter_t` \*Iterator, int8\_t Value)
- LBMSDMEpDLL int `lbmsdm_iter_set_uint8` (`lbmsdm_iter_t` \*Iterator, uint8\_t Value)
- LBMSDMEpDLL int `lbmsdm_iter_set_int16` (`lbmsdm_iter_t` \*Iterator, int16\_t Value)
- LBMSDMEpDLL int `lbmsdm_iter_set_uint16` (`lbmsdm_iter_t` \*Iterator, uint16\_t Value)
- LBMSDMEpDLL int `lbmsdm_iter_set_int32` (`lbmsdm_iter_t` \*Iterator, int32\_t Value)
- LBMSDMEpDLL int `lbmsdm_iter_set_uint32` (`lbmsdm_iter_t` \*Iterator, uint32\_t Value)
- LBMSDMEpDLL int `lbmsdm_iter_set_int64` (`lbmsdm_iter_t` \*Iterator, int64\_t Value)
- LBMSDMEpDLL int `lbmsdm_iter_set_uint64` (`lbmsdm_iter_t` \*Iterator, uint64\_t Value)
- LBMSDMEpDLL int `lbmsdm_iter_set_float` (`lbmsdm_iter_t` \*Iterator, float Value)
- LBMSDMEpDLL int `lbmsdm_iter_set_double` (`lbmsdm_iter_t` \*Iterator, double Value)
- LBMSDMEpDLL int `lbmsdm_iter_set_decimal` (`lbmsdm_iter_t` \*Iterator, const `lbmsdm_decimal_t` \*Value)
- LBMSDMEpDLL int `lbmsdm_iter_set_timestamp` (`lbmsdm_iter_t` \*Iterator, const struct timeval \*Value)
- LBMSDMEpDLL int `lbmsdm_iter_set_message` (`lbmsdm_iter_t` \*Iterator, const `lbmsdm_msg_t` \*Value)
- LBMSDMEpDLL int `lbmsdm_iter_set_string` (`lbmsdm_iter_t` \*Iterator, const char \*Value)
- LBMSDMEpDLL 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.*

- LBMSDMEpDLL 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 `LBMSDMEExpDLL 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 `LBMSDMEExpDLL 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 `LBMSDMEExpDLL 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

- LBMSDMEpDLL 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.*
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_int8\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_uint8\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_int16\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_uint16\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_int32\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_uint32\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_int64\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_uint64\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_float\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_double\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_decimal\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_timestamp\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_message\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_string\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_unicode\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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

- `LBMSDMEpDLL 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.*

- `LBMSDMEpDLL int lbmsdm_msg_set_int8_array_name (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_set_uint8_array_name (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_set_int16_array_name (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_set_uint16_array_name (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_set_int32_array_name (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_set_uint32_array_name (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_set_int64_array_name (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_set_uint64_array_name (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_set_float_array_name (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_set_double_array_name (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_set_decimal_array_name (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_set_timestamp_array_name (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_set_message_array_name (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_set_string_array_name (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL int lbmsdm_msg_set_unicode_array_name (lbmsdm_msg_t *Message, const char *Name)`
- `LBMSDMEpDLL 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 `LBMSDMEExpDLL 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 `LBMSDMEExpDLL 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 `LBMSDMEExpDLL 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 `LBMSDMEExpDLL 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 `LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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**

- `LBMSDMEExpDLL int lbmsdm_iter_set_boolean_array (lbmsdm_iter_t *Iterator)`  
*Set a field in a message by field name to an array field.*
- `LBMSDMEExpDLL int lbmsdm_iter_set_int8_array (lbmsdm_iter_t *Iterator)`
- `LBMSDMEExpDLL int lbmsdm_iter_set_uint8_array (lbmsdm_iter_t *Iterator)`
- `LBMSDMEExpDLL int lbmsdm_iter_set_int16_array (lbmsdm_iter_t *Iterator)`
- `LBMSDMEExpDLL int lbmsdm_iter_set_uint16_array (lbmsdm_iter_t *Iterator)`
- `LBMSDMEExpDLL int lbmsdm_iter_set_int32_array (lbmsdm_iter_t *Iterator)`
- `LBMSDMEExpDLL int lbmsdm_iter_set_uint32_array (lbmsdm_iter_t *Iterator)`
- `LBMSDMEExpDLL int lbmsdm_iter_set_int64_array (lbmsdm_iter_t *Iterator)`
- `LBMSDMEExpDLL int lbmsdm_iter_set_uint64_array (lbmsdm_iter_t *Iterator)`
- `LBMSDMEExpDLL int lbmsdm_iter_set_float_array (lbmsdm_iter_t *Iterator)`
- `LBMSDMEExpDLL int lbmsdm_iter_set_double_array (lbmsdm_iter_t *Iterator)`
- `LBMSDMEExpDLL int lbmsdm_iter_set_decimal_array (lbmsdm_iter_t *Iterator)`
- `LBMSDMEExpDLL int lbmsdm_iter_set_timestamp_array (lbmsdm_iter_t *Iterator)`
- `LBMSDMEExpDLL int lbmsdm_iter_set_message_array (lbmsdm_iter_t *Iterator)`
- `LBMSDMEExpDLL int lbmsdm_iter_set_string_array (lbmsdm_iter_t *Iterator)`
- `LBMSDMEExpDLL int lbmsdm_iter_set_unicode_array (lbmsdm_iter_t *Iterator)`
- `LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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** **LBMSDMExpDLL** **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** **LBMSDMExpDLL** **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** **LBMSDMExpDLL** **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** **LBMSDMExpDLL** **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** **LBMSDMExpDLL** **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** **LBMSDMExpDLL** **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

- LBMSDMEpDLL 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.*

- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_int8\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, int8\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_uint8\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, uint8\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_int16\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, int16\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_uint16\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, uint16\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_int32\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, int32\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_uint32\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, uint32\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_int64\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, int64\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_uint64\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, uint64\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_float\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, float Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_double\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, double Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_decimal\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, const lbmsdm\_decimal\_t \*Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_timestamp\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, const struct timeval \*Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_message\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, const lbmsdm\_msg\_t \*Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_string\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, const char \*Value)
- LBMSDMEpDLL 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.*

- `LBMSDMEpDLL 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** `LBMSDMEpDLL 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** `LBMSDMEpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** `LBMSDMExpDLL 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** `LBMSDMExpDLL 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** `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`)

**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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 `void* lbm_apphdr_chain_elem_t_stct::data`

Pointer to the app header data

##### 7.1.1.2 `size_t lbm_apphdr_chain_elem_t_stct::len`

length of the data pointed to by `data`

**7.1.1.3 `lbm_ushort_t lbm_apphdr_chain_elem_t_stct::subtype`**

Code representing the subtype (if any) of the data in this element.

**7.1.1.4 `lbm_uchar_t lbm_apphdr_chain_elem_t_stct::type`**

Code representing the type of data in this element.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.2 `lbm_async_operation_func_t` Struct Reference

Structure that holds information for asynchronous operation callbacks.

```
#include <lbm.h>
```

### Data Fields

- [lbm\\_async\\_operation\\_function\\_cb](#) `func`
- `lbm_event_queue_t * evq`
- `void * clientd`
- `int flags`

### 7.2.1 Field Documentation

#### 7.2.1.1 `void* lbm_async_operation_func_t::clientd`

A client object pointer to be passed back in to the specified callback.

#### 7.2.1.2 `lbm_event_queue_t* lbm_async_operation_func_t::evq`

An event queue pointer; not yet supported. Should be set to NULL.

#### 7.2.1.3 `int lbm_async_operation_func_t::flags`

Flags that indicate which optional portions are included and may affect callback behavior.

#### 7.2.1.4 `lbm_async_operation_function_cb lbm_async_operation_func_t::func`

A callback function to receive status and completion of an asynchronous operation.

The documentation for this struct was generated from the following file:

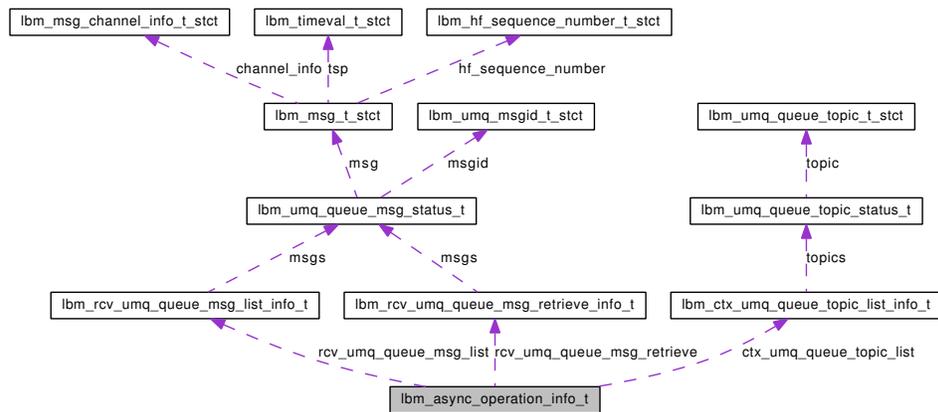
- [lbm.h](#)

### 7.3 lbm\_async\_operation\_info\_t Struct Reference

Results struct returned via the user-specified asynchronous operation callback from any asynchronous API.

```
#include <lbm.h>
```

Collaboration diagram for `lbm_async_operation_info_t`:



#### Data Fields

- `int type`
- `int status`
- `int flags`
- `lbm_async_operation_handle_t handle`
- union {
  - `lbm_ctx_umq_queue_topic_list_info_t * ctx_umq_queue_topic_list`
  - `lbm_rcv_umq_queue_msg_list_info_t * rcv_umq_queue_msg_list`
  - `lbm_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 lbm_async_operation_info_t::flags`

Flags with extra information about the async operation.

### 7.3.2.2 `lbm_async_operation_handle_t lbm_async_operation_info_t::handle`

An opaque handle to the asynchronous operation.

### 7.3.2.3 `union { ... } lbm_async_operation_info_t::info`

Operation-specific results.

### 7.3.2.4 `int lbm_async_operation_info_t::status`

The current status of the operation.

### 7.3.2.5 `int lbm_async_operation_info_t::type`

The type of asynchronous operation.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.4 `lbm_context_event_func_t_stct` Struct Reference

Structure that holds the application callback for context-level events.

```
#include <lbm.h>
```

### Data Fields

- `lbm_context_event_cb_proc` **func**
- `lbm_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:

- [lbm.h](#)

## 7.5 lbm\_context\_event\_umq\_registration\_complete\_ex\_t\_stct Struct Reference

### 7.5 lbm\_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 <lbm.h>
```

#### Data Fields

- int [flags](#)
- [lbm\\_umq\\_regid\\_t](#) registration\_id
- [lbm\\_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 [lbm\\_context\\_event\\_umq\\_registration\\_complete\\_ex\\_t\\_stct::flags](#)

Flags that indicate which optional portions are included

##### 7.5.2.2 char [lbm\\_context\\_event\\_umq\\_registration\\_complete\\_ex\\_t\\_stct::queue](#)[LBM\_UMQ\_MAX\_QUEUE\_STRLEN]

The name of the queue registered with

##### 7.5.2.3 [lbm\\_uint\\_t](#) [lbm\\_context\\_event\\_umq\\_registration\\_complete\\_ex\\_t\\_stct::queue\\_id](#)

The Queue ID of the queue

##### 7.5.2.4 [lbm\\_umq\\_regid\\_t](#) [lbm\\_context\\_event\\_umq\\_registration\\_complete\\_ex\\_t\\_stct::registration\\_id](#)

Registration ID used for the registration

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.6 `lbm_context_event_umq_registration_ex_t_stct` Struct Reference

Structure that holds queue registration information for the UMQ context in an extended form.

```
#include <lbm.h>
```

### Data Fields

- `int flags`
- `lbm_umq_regid_t registration_id`
- `lbm_uint_t queue_id`
- `lbm_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 lbm_context_event_umq_registration_ex_t_stct::flags`

Flags that indicate which optional portions are included

#### 7.6.2.2 `char lbm_context_event_umq_registration_ex_t_stct::queue[LBM_UMQ_MAX_QUEUE_STRLEN]`

The name of the queue registered with

#### 7.6.2.3 `lbm_uint_t lbm_context_event_umq_registration_ex_t_stct::queue_id`

The Queue ID of the queue

#### 7.6.2.4 `char lbm_context_event_umq_registration_ex_t_stct::queue_instance[LBM_UME_MAX_STORE_STRLEN]`

The instance of the queue registered with

#### 7.6.2.5 [lbm\\_uint\\_t lbm\\_context\\_event\\_umq\\_registration\\_ex\\_t\\_stct::queue\\_instance\\_index](#)

The index of the instance of the queue registered with

#### 7.6.2.6 [lbm\\_umq\\_regid\\_t lbm\\_context\\_event\\_umq\\_registration\\_ex\\_t\\_stct::registration\\_id](#)

Registration ID used for the registration

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.7 `lbm_context_rcv_immediate_msgs_func_t_stct` Struct Reference

Structure that holds the application callback for receiving topic-less immediate mode messages.

```
#include <lbm.h>
```

### Data Fields

- `lbm_immediate_msg_cb_proc` **func**
- `lbm_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 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.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.8 `lbm_context_src_event_func_t_stct` Struct Reference

Structure that holds the application callback for context-level source events.

```
#include <lbm.h>
```

### Data Fields

- [lbm\\_context\\_src\\_cb\\_proc](#) `func`
- `lbm_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:

- [lbm.h](#)

## 7.9 lbm\_context\_stats\_t\_stct Struct Reference

Structure that holds statistics for a context.

```
#include <lbm.h>
```

### Data Fields

- lbm\_ulong\_t [tr\\_dgrams\\_sent](#)
- lbm\_ulong\_t [tr\\_bytes\\_sent](#)
- lbm\_ulong\_t [tr\\_dgrams\\_rcved](#)
- lbm\_ulong\_t [tr\\_bytes\\_rcved](#)
- lbm\_ulong\_t [tr\\_dgrams\\_dropped\\_ver](#)
- lbm\_ulong\_t [tr\\_dgrams\\_dropped\\_type](#)
- lbm\_ulong\_t [tr\\_dgrams\\_dropped\\_malformed](#)
- lbm\_ulong\_t [tr\\_dgrams\\_send\\_failed](#)
- lbm\_ulong\_t [tr\\_src\\_topics](#)
- lbm\_ulong\_t [tr\\_rcv\\_topics](#)
- lbm\_ulong\_t [tr\\_rcv\\_unresolved\\_topics](#)
- lbm\_ulong\_t [lbtrm\\_unknown\\_msgs\\_rcved](#)
- lbm\_ulong\_t [lbtru\\_unknown\\_msgs\\_rcved](#)
- lbm\_ulong\_t [send\\_blocked](#)
- lbm\_ulong\_t [send\\_would\\_block](#)
- lbm\_ulong\_t [resp\\_blocked](#)
- lbm\_ulong\_t [resp\\_would\\_block](#)
- lbm\_ulong\_t [uim\\_dup\\_msgs\\_rcved](#)
- lbm\_ulong\_t [uim\\_msgs\\_no\\_stream\\_rcved](#)
- lbm\_ulong\_t [fragments\\_lost](#)
- lbm\_ulong\_t [fragments\\_unrecoverably\\_lost](#)
- lbm\_ulong\_t [rcv\\_cb\\_svc\\_time\\_min](#)
- lbm\_ulong\_t [rcv\\_cb\\_svc\\_time\\_max](#)
- lbm\_ulong\_t [rcv\\_cb\\_svc\\_time\\_mean](#)

### 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 `lbm_ulong_t lbm_context_stats_t_stct::fragments_lost`

For internal use only. Number of data message fragments detected as lost in the context. UM fragments messages larger than the maximum datagram size for the transport and assigns a unique sequence number to each fragment. UM increments the count when a delivery controller in the context detects a gap in fragment sequence numbers. Lost request responses, Multicast Immediate Message (MIM) messages, or Unicast Immediate Message (UIM) control messages do not increase this statistic. Lost message fragments sent to hot-failover receivers with arrival-order delivery and lost message fragments sent over LBT-SMX do not increase this statistic.

### 7.9.2.2 `lbm_ulong_t lbm_context_stats_t_stct::fragments_unrecoverably_lost`

For internal use only. Number of data message fragments detected as unrecoverably lost in the context. UM fragments messages larger than the maximum datagram size for the transport and assigns a unique sequence number to each fragment. UM increments the count when a delivery controller in the context sends a `LBM_MSG_UNRECOVERABLE_LOSS` or `LBM_MSG_UNRECOVERABLE_LOSS_BURST` event to the receiving application. Unrecoverably lost message fragments sent to hot-failover receivers with arrival-order delivery do not increment this statistic. An unrecoverable loss event sent by a delivery controller for a receiver underlying the hot-failover receiver does increment this statistic if another underlying receiver was able to compensate. Unrecoverably lost message fragments sent over LBT-SMX do not increment this statistic.

### 7.9.2.3 `lbm_ulong_t lbm_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.

### 7.9.2.4 `lbm_ulong_t lbm_context_stats_t_stct::lbtru_unknown_msgs_rcved`

Number of datagrams received that do not belong to a currently-subscribed LBT-RM transport session. This is typically due to improper or suboptimal UM configuration, or to interference from non-UM network applications. Steady increases in this statistic indicate a waste in CPU resources to read and discard the datagrams. A common cause is different application instances creating transport sessions with the same multicast group and destination port.

### 7.9.2.5 lbm\_ulong\_t lbm\_context\_stats\_t\_stct::rcv\_cb\_svc\_time\_max

For internal use only. The maximum time in microseconds used to call user receive callbacks of type [lbm\\_rcv\\_cb\\_proc\(\)](#) for wildcard, hot-failover, and regular receivers. Does not include SMX receivers. Does not include minimum times for any other context thread callbacks, such as timer callback or immediate message callbacks. The initial value before any data has been reported is zero. UM computes the maximum value as the larger of the current measured value and the current statistic value. A value of zero equates to no measurement. For the Java and .NET APIs, includes the overhead time spent crossing the managed/jni boundaries. You must set the configuration option `receiver_callback_service_time_enabled` to populate this statistic.

### 7.9.2.6 lbm\_ulong\_t lbm\_context\_stats\_t\_stct::rcv\_cb\_svc\_time\_mean

For internal use only. The mean time in microseconds used to call user receive callbacks of type [lbm\\_rcv\\_cb\\_proc\(\)](#) for wildcard, hot-failover, and regular receivers. Does not include SMX receivers. Does not include minimum times for any other context thread callbacks, such as timer callback or immediate message callbacks. For the Java and .NET APIs, includes the overhead time spent crossing the managed/jni boundaries. You must set the configuration option `receiver_callback_service_time_enabled` to populate this statistic.

### 7.9.2.7 lbm\_ulong\_t lbm\_context\_stats\_t\_stct::rcv\_cb\_svc\_time\_min

For internal use only. The minimum time in microseconds used to call user receive callbacks of type [lbm\\_rcv\\_cb\\_proc\(\)](#) for wildcard, hot-failover, and regular receivers. Does not include SMX receivers. Does not include minimum times for any other context thread callbacks, such as timer callback or immediate message callbacks. The initial value before UM Monitoring reports any data is the largest possible unsigned integer, which is 4294967295 for 32-bit system and 18446744073709551615 for 64-bit system. Ultra Messaging computes the minimum value as the smaller of the current measured value and the current statistic value. The largest possible value equates to no measurement. For the Java and .NET APIs, includes the overhead time spent crossing the managed/jni boundaries. You must set the configuration option `receiver_callback_service_time_enabled` to populate this statistic.

### 7.9.2.8 lbm\_ulong\_t lbm\_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.

**7.9.2.9 `lbm_ulong_t lbm_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.

**7.9.2.10 `lbm_ulong_t lbm_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.

**7.9.2.11 `lbm_ulong_t lbm_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.

**7.9.2.12 `lbm_ulong_t lbm_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.

**7.9.2.13 `lbm_ulong_t lbm_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.

**7.9.2.14 `lbm_ulong_t lbm_context_stats_t_stct::tr_dgrams_dropped_malformed`**

Number of topic resolution datagrams discarded due to being malformed or corrupted.

**7.9.2.15 `lbm_ulong_t lbm_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.

**7.9.2.16** `lbm_ulong_t lbm_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.

**7.9.2.17** `lbm_ulong_t lbm_context_stats_t_stct::tr_dgrams_reved`

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.

**7.9.2.18** `lbm_ulong_t lbm_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.

**7.9.2.19** `lbm_ulong_t lbm_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.

**7.9.2.20** `lbm_ulong_t lbm_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.

**7.9.2.21** `lbm_ulong_t lbm_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.

**7.9.2.22** `lbm_ulong_t lbm_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.

**7.9.2.23** `lbm_ulong_t lbm_context_stats_t_stct::uim_dup_msgs_rcved`

Number of duplicate unicast immediate messages (UIMs) received and dropped.

**7.9.2.24** `lbm_ulong_t lbm_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:

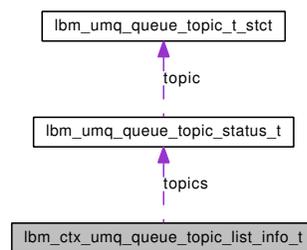
- [lbm.h](#)

## 7.10 lbm\_ctx\_umq\_queue\_topic\_list\_info\_t Struct Reference

Struct containing an array of queue topics retrieved via `lbm_umq_queue_topic_list`.

```
#include <lbm.h>
```

Collaboration diagram for `lbm_ctx_umq_queue_topic_list_info_t`:



### Data Fields

- `lbm_umq_queue_topic_status_t * topics`
- `int num_topics`

#### 7.10.1 Field Documentation

##### 7.10.1.1 `int lbm_ctx_umq_queue_topic_list_info_t::num_topics`

The length, in number of elements, of the topic objects array.

##### 7.10.1.2 `lbm_umq_queue_topic_status_t * lbm_ctx_umq_queue_topic_list_info_t::topics`

An array of UMQ topic status objects.

The documentation for this struct was generated from the following file:

- `lbm.h`

## 7.11 `lbm_delete_cb_info_t_stct` Struct Reference

Structure passed to the `lbm_hypertopic_rcv_delete()` function so that a deletion callback may be called.

```
#include <lbmht.h>
```

### Data Fields

- `lbm_delete_cb_proc` `cbproc`
- `void *` `clientd`

#### 7.11.1 Field Documentation

##### 7.11.1.1 `lbm_delete_cb_proc` `lbm_delete_cb_info_t_stct::cbproc`

The cancel callback function

##### 7.11.1.2 `void*` `lbm_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:

- `lbmht.h`

## 7.12 lbm\_event\_queue\_cancel\_cb\_info\_t\_stct Struct Reference

Structure passed to cancel/delete functions so that a cancel callback may be called.

```
#include <lbm.h>
```

### Data Fields

- [lbm\\_event\\_queue\\_t \\* event\\_queue](#)
- [lbm\\_event\\_queue\\_cancel\\_cb\\_proc cbproc](#)
- [void \\* clientd](#)

#### 7.12.1 Field Documentation

##### 7.12.1.1 [lbm\\_event\\_queue\\_cancel\\_cb\\_proc lbm\\_event\\_queue\\_cancel\\_cb\\_info\\_t\\_stct::cbproc](#)

The cancel callback function

##### 7.12.1.2 [void\\* lbm\\_event\\_queue\\_cancel\\_cb\\_info\\_t\\_stct::clientd](#)

Client Data passed in the cancel callback when called

##### 7.12.1.3 [lbm\\_event\\_queue\\_t\\* lbm\\_event\\_queue\\_cancel\\_cb\\_info\\_t\\_stct::event\\_queue](#)

The event queue for the cancel callback

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.13 `lbm_event_queue_stats_t_stct` Struct Reference

Structure that holds statistics for an event queue.

```
#include <lbm.h>
```

### Data Fields

- `lbm_ulong_t data_msgs`
- `lbm_ulong_t data_msgs_tot`
- `lbm_ulong_t data_msgs_svc_min`
- `lbm_ulong_t data_msgs_svc_mean`
- `lbm_ulong_t data_msgs_svc_max`
- `lbm_ulong_t resp_msgs`
- `lbm_ulong_t resp_msgs_tot`
- `lbm_ulong_t resp_msgs_svc_min`
- `lbm_ulong_t resp_msgs_svc_mean`
- `lbm_ulong_t resp_msgs_svc_max`
- `lbm_ulong_t topicless_im_msgs`
- `lbm_ulong_t topicless_im_msgs_tot`
- `lbm_ulong_t topicless_im_msgs_svc_min`
- `lbm_ulong_t topicless_im_msgs_svc_mean`
- `lbm_ulong_t topicless_im_msgs_svc_max`
- `lbm_ulong_t wrcv_msgs`
- `lbm_ulong_t wrcv_msgs_tot`
- `lbm_ulong_t wrcv_msgs_svc_min`
- `lbm_ulong_t wrcv_msgs_svc_mean`
- `lbm_ulong_t wrcv_msgs_svc_max`
- `lbm_ulong_t io_events`
- `lbm_ulong_t io_events_tot`
- `lbm_ulong_t io_events_svc_min`
- `lbm_ulong_t io_events_svc_mean`
- `lbm_ulong_t io_events_svc_max`
- `lbm_ulong_t timer_events`
- `lbm_ulong_t timer_events_tot`
- `lbm_ulong_t timer_events_svc_min`
- `lbm_ulong_t timer_events_svc_mean`
- `lbm_ulong_t timer_events_svc_max`
- `lbm_ulong_t source_events`
- `lbm_ulong_t source_events_tot`
- `lbm_ulong_t source_events_svc_min`
- `lbm_ulong_t source_events_svc_mean`
- `lbm_ulong_t source_events_svc_max`

- lbm\_ulong\_t [unblock\\_events](#)
- lbm\_ulong\_t [unblock\\_events\\_tot](#)
- lbm\_ulong\_t [cancel\\_events](#)
- lbm\_ulong\_t [cancel\\_events\\_tot](#)
- lbm\_ulong\_t [cancel\\_events\\_svc\\_min](#)
- lbm\_ulong\_t [cancel\\_events\\_svc\\_mean](#)
- lbm\_ulong\_t [cancel\\_events\\_svc\\_max](#)
- lbm\_ulong\_t [context\\_source\\_events](#)
- lbm\_ulong\_t [context\\_source\\_events\\_tot](#)
- lbm\_ulong\_t [context\\_source\\_events\\_svc\\_min](#)
- lbm\_ulong\_t [context\\_source\\_events\\_svc\\_mean](#)
- lbm\_ulong\_t [context\\_source\\_events\\_svc\\_max](#)
- lbm\_ulong\_t [events](#)
- lbm\_ulong\_t [events\\_tot](#)
- lbm\_ulong\_t [age\\_min](#)
- lbm\_ulong\_t [age\\_mean](#)
- lbm\_ulong\_t [age\\_max](#)
- lbm\_ulong\_t [callback\\_events](#)
- lbm\_ulong\_t [callback\\_events\\_tot](#)
- lbm\_ulong\_t [callback\\_events\\_svc\\_min](#)
- lbm\_ulong\_t [callback\\_events\\_svc\\_mean](#)
- lbm\_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 lbm\_ulong\_t lbm\_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 enqueueing until de-queueing. Configuration option `queue_age_enabled` must be activated.

**7.13.2.2 `lbm_ulong_t lbm_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.

**7.13.2.3 `lbm_ulong_t lbm_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 enqueueement until de-queueement. Configuration option `queue_age_enabled` must be activated.

**7.13.2.4 `lbm_ulong_t lbm_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.

**7.13.2.5 `lbm_ulong_t lbm_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-queueement until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.6 `lbm_ulong_t lbm_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.

**7.13.2.7 `lbm_ulong_t lbm_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.

**7.13.2.8 `lbm_ulong_t lbm_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.

**7.13.2.9 `lbm_ulong_t lbm_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.

**7.13.2.10 `lbm_ulong_t lbm_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.

**7.13.2.11 `lbm_ulong_t lbm_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.

**7.13.2.12 `lbm_ulong_t lbm_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.

**7.13.2.13 `lbm_ulong_t lbm_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.

**7.13.2.14 `lbm_ulong_t lbm_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.

**7.13.2.15** `lbm_ulong_t lbm_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-queuement until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.16** `lbm_ulong_t lbm_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-queuement until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.17** `lbm_ulong_t lbm_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-queuement until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.18** `lbm_ulong_t lbm_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.

**7.13.2.19** `lbm_ulong_t lbm_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.

**7.13.2.20** `lbm_ulong_t lbm_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-queuement until the application has finished servicing the message. Configuration option `queue_service_time_enabled` must be activated.

#### 7.13.2.21 `lbm_ulong_t lbm_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-queuement until the application has finished servicing the message. Configuration option `queue_service_time_enabled` must be activated.

#### 7.13.2.22 `lbm_ulong_t lbm_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-queuement until the application has finished servicing the message. Configuration option `queue_service_time_enabled` must be activated.

#### 7.13.2.23 `lbm_ulong_t lbm_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.

#### 7.13.2.24 `lbm_ulong_t lbm_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.

#### 7.13.2.25 `lbm_ulong_t lbm_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.

#### 7.13.2.26 `lbm_ulong_t lbm_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.

**7.13.2.27 `lbm_ulong_t lbm_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-queuement until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.28 `lbm_ulong_t lbm_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-queuement until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.29 `lbm_ulong_t lbm_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-queuement until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.30 `lbm_ulong_t lbm_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.

**7.13.2.31 `lbm_ulong_t lbm_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.

**7.13.2.32 `lbm_ulong_t lbm_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-queuement until the application has finished servicing the message. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.33 [lbm\\_ulong\\_t lbm\\_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-queuement until the application has finished servicing the message. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.34 [lbm\\_ulong\\_t lbm\\_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-queuement until the application has finished servicing the message. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.35 [lbm\\_ulong\\_t lbm\\_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.

**7.13.2.36 [lbm\\_ulong\\_t lbm\\_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.

**7.13.2.37 [lbm\\_ulong\\_t lbm\\_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-queuement until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.38 [lbm\\_ulong\\_t lbm\\_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-queuement until the application has finished servicing the message. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.39 `lbm_ulong_t lbm_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-queuement until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.40 `lbm_ulong_t lbm_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.

**7.13.2.41 `lbm_ulong_t lbm_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.

**7.13.2.42 `lbm_ulong_t lbm_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-queuement until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.43 `lbm_ulong_t lbm_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-queuement until the application has finished servicing the message. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.44 `lbm_ulong_t lbm_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-queuement until the application has finished servicing the event. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.45** `lbm_ulong_t lbm_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.

**7.13.2.46** `lbm_ulong_t lbm_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.

**7.13.2.47** `lbm_ulong_t lbm_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.

**7.13.2.48** `lbm_ulong_t lbm_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.

**7.13.2.49** `lbm_ulong_t lbm_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-queuement until the application has finished servicing the message. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.50 `lbm_ulong_t lbm_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.

**7.13.2.51 `lbm_ulong_t lbm_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.

**7.13.2.52 `lbm_ulong_t lbm_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.

**7.13.2.53 `lbm_ulong_t lbm_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.

**7.13.2.54 `lbm_ulong_t lbm_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-queuement until the application has finished servicing the message. Configuration option `queue_service_time_enabled` must be activated.

**7.13.2.55 `lbm_ulong_t lbm_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.

**7.13.2.56 `lbm_ulong_t lbm_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 du-

rations measured from the point of de-queuement until the application has finished servicing the message. Configuration option `queue_service_time_enabled` must be activated.

#### 7.13.2.57 `lbm_ulong_t lbm_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.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.14 `lbm_flight_size_inflight_t_stct` Struct Reference

Structure that holds information for source total inflight messages and bytes.

```
#include <lbm.h>
```

### Data Fields

- `int` `messages`
- `lbm_uint64_t` `bytes`

#### 7.14.1 Field Documentation

##### 7.14.1.1 `lbm_uint64_t lbm_flight_size_inflight_t_stct::bytes`

Current amount of inflight payload bytes

##### 7.14.1.2 `int lbm_flight_size_inflight_t_stct::messages`

Current amount of inflight messages

The documentation for this struct was generated from the following file:

- `lbm.h`

## 7.15 lbm\_hf\_sequence\_number\_t\_stct Union Reference

Structure to hold a hot failover sequence number.

```
#include <lbm.h>
```

### Data Fields

- `lbm_uint32_t u32`
- `lbm_uint64_t u64`

#### 7.15.1 Field Documentation

##### 7.15.1.1 `lbm_uint32_t lbm_hf_sequence_number_t_stct::u32`

32 bit hot failover sequence number

##### 7.15.1.2 `lbm_uint64_t lbm_hf_sequence_number_t_stct::u64`

64 bit hot failover sequence number

The documentation for this union was generated from the following file:

- [lbm.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 `lbm_ipv4_address_mask_t_stct` Struct Reference

Structure that holds an IPv4 address and a CIDR style netmask.

```
#include <lbm.h>
```

### Data Fields

- `lbm_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 `lbm_uint_t lbm_ipv4_address_mask_t_stct::addr`

IPv4 address

##### 7.17.2.2 `int lbm_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:

- `lbm.h`

## 7.18 `lbm_mim_unrecloss_func_t_stct` Struct Reference

Structure that holds the application callback for multicast immediate message unrecoverable loss notification.

```
#include <lbm.h>
```

### Data Fields

- `lbm_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:

- [lbm.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 lbm\_uint32\_t [lbm\\_msg\\_channel\\_info\\_t\\_stct::channel\\_number](#)

Channel number in the range 0-4294967295

##### 7.19.2.2 int [lbm\\_msg\\_channel\\_info\\_t\\_stct::flags](#)

Channel flags

- LBM\_MSG\_FLAG\_NUMBERED\_CHANNEL Message was delivered on a numbered channel.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.20 `lbm_msg_fragment_info_t_stct` Struct Reference

Structure that holds fragment information for UM messages when appropriate.

```
#include <lbm.h>
```

### Data Fields

- `lbm_uint_t start_sequence_number`
- `lbm_uint_t offset`
- `lbm_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 `lbm_msg_retrieve_fragment_info()`.

#### 7.20.2 Field Documentation

##### 7.20.2.1 `lbm_uint_t lbm_msg_fragment_info_t_stct::offset`

The offset (in bytes) of this fragment from the message beginning

##### 7.20.2.2 `lbm_uint_t lbm_msg_fragment_info_t_stct::start_sequence_number`

The sequence number of the fragment that starts the message

##### 7.20.2.3 `lbm_uint_t lbm_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:

- `lbm.h`

## 7.21 lbm\_msg\_gateway\_info\_t\_stct Struct Reference

Structure that holds originating information for UM messages which arrived via a gateway.

```
#include <lbm.h>
```

### Data Fields

- lbm\_uint\_t [sequence\\_number](#)
- char [source](#) [LBM\_MSG\_MAX\_SOURCE\_LEN]

#### 7.21.1 Detailed Description

**Deprecated**

#### 7.21.2 Field Documentation

##### 7.21.2.1 lbm\_uint\_t [lbm\\_msg\\_gateway\\_info\\_t\\_stct::sequence\\_number](#)

The original sequence number (relative to the original transport.)

##### 7.21.2.2 char [lbm\\_msg\\_gateway\\_info\\_t\\_stct::source](#)[LBM\_MSG\_MAX\_SOURCE\_LEN]

The original source string.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.22 `lbm_msg_properties_iter_t_stct` Struct Reference

A struct used for iterating over properties pointed to by an `lbm_msg_properties_t`.

```
#include <lbm.h>
```

### Data Fields

- `const char * name`
- `char * data`
- `size_t size`
- `int type`

The documentation for this struct was generated from the following file:

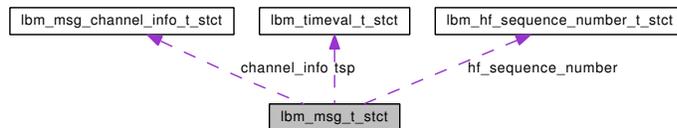
- [lbm.h](#)

## 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\\_msgid](#)
- 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 `lbm_msg_channel_info_t*` `lbm_msg_t_stct::channel_info`

Channel information set when using Spectrum channels

#### 7.23.1.2 `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!)

#### 7.23.1.3 `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.

#### 7.23.1.4 `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

#### 7.23.1.5 `lbm_hf_sequence_number_t` `lbm_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`.

**7.23.1.6** `size_t lbm_msg_t_stct::len`

Length of data in bytes

**7.23.1.7** `lbm_msg_properties_t* lbm_msg_t_stct::properties`

Message properties structure for this message

**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** `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](#).

**7.23.1.11** `const void* lbm_msg_t_stct::source_clientd`

Pointer set by lbm\_rcv\_src\_notification\_create\_function\_cb callback

**7.23.1.12** `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.

**7.23.1.13** `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.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.24 `lbm_msg_ume_deregistration_ex_t_stct` Struct Reference

Structure that holds store deregistration information for the UM receiver in an extended form.

```
#include <lbm.h>
```

### Data Fields

- `int flags`
- `lbm_uint_t src_registration_id`
- `lbm_uint_t rcv_registration_id`
- `lbm_uint_t sequence_number`
- `lbm_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 lbm_msg_ume_deregistration_ex_t_stct::flags`

Flags

#### 7.24.2.2 `lbm_uint_t lbm_msg_ume_deregistration_ex_t_stct::rcv_registration_id`

The registration ID for the receiver

#### 7.24.2.3 `lbm_uint_t lbm_msg_ume_deregistration_ex_t_stct::sequence_number`

The sequence number for the receiver to end at as reported by the store

**7.24.2.4** `lbm_uint_t lbm_msg_ume_deregistration_ex_t_stct::src_registration_id`

The registration ID for the source

**7.24.2.5** `char lbm_msg_ume_deregistration_ex_t_stct::store[LBM_UME_MAX_STORE_STRLEN]`

The store that was registered with

**7.24.2.6** `lbm_ushort_t lbm_msg_ume_deregistration_ex_t_stct::store_index`

The store index of the store involved

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.25 `lbm_msg_ume_registration_complete_ex_t_stct` Struct Reference

Structure that holds information for receivers after registration is complete to all involved stores.

```
#include <lbm.h>
```

### Data Fields

- `int flags`
- `lbm_uint_t sequence_number`
- `lbm_uint64_t src_session_id`

#### 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 lbm_msg_ume_registration_complete_ex_t_stct::flags`

Flags

##### 7.25.2.2 `lbm_uint_t lbm_msg_ume_registration_complete_ex_t_stct::sequence_number`

The sequence number that will be the first requested

##### 7.25.2.3 `lbm_uint64_t lbm_msg_ume_registration_complete_ex_t_stct::src_session_id`

The session ID for the source

The documentation for this struct was generated from the following file:

- `lbm.h`

## 7.26 `lbm_msg_ume_registration_ex_t_stct` Struct Reference

Structure that holds store registration information for the UM receiver in an extended form.

```
#include <lbm.h>
```

### Data Fields

- `int flags`
- `lbm_uint_t src_registration_id`
- `lbm_uint_t rcv_registration_id`
- `lbm_uint_t sequence_number`
- `lbm_ushort_t store_index`
- `char store` [LBM\_UME\_MAX\_STORE\_STRLEN]
- `lbm_uint64_t src_session_id`

### 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 lbm_msg_ume_registration_ex_t_stct::flags`

Flags

#### 7.26.2.2 `lbm_uint_t lbm_msg_ume_registration_ex_t_stct::rcv_registration_id`

The registration ID for the receiver

#### 7.26.2.3 `lbm_uint_t lbm_msg_ume_registration_ex_t_stct::sequence_number`

The sequence number for the receiver to start at as reported by the store

#### 7.26.2.4 `lbm_uint_t lbm_msg_ume_registration_ex_t_stct::src_registration_id`

The registration ID for the source

**7.26.2.5** `lbm_uint64_t lbm_msg_ume_registration_ex_t_stct::src_session_id`

The session ID for the source

**7.26.2.6** `char lbm_msg_ume_registration_ex_t_stct::store[LBM_UME_MAX_STORE_STRLEN]`

The store that was registered with

**7.26.2.7** `lbm_ushort_t lbm_msg_ume_registration_ex_t_stct::store_index`

The store index of the store involved

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.27 `lbm_msg_ume_registration_t_stct` Struct Reference

Structure that holds store registration information for the UMP receiver.

```
#include <lbm.h>
```

### Data Fields

- `lbm_uint_t src_registration_id`
- `lbm_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 `lbm_uint_t lbm_msg_ume_registration_t_stct::rcv_registration_id`

The registration ID for the receiver

##### 7.27.2.2 `lbm_uint_t lbm_msg_ume_registration_t_stct::src_registration_id`

The registration ID for the source

The documentation for this struct was generated from the following file:

- `lbm.h`

## 7.28 `lbm_msg_umq_deregistration_complete_ex_t_stct` Struct Reference

Structure that holds information for receivers after they de-register from a queue.

```
#include <lbm.h>
```

### Data Fields

- `int flags`
- `lbm_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 lbm_msg_umq_deregistration_complete_ex_t_stct::flags`

Flags that indicate which optional portions are included

#### 7.28.2.2 `lbm_uint_t lbm_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:

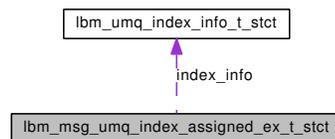
- `lbm.h`

## 7.29 lbm\_msg\_umq\_index\_assigned\_ex\_t\_stct Struct Reference

Structure that holds beginning-of-index information for receivers.

```
#include <lbm.h>
```

Collaboration diagram for lbm\_msg\_umq\_index\_assigned\_ex\_t\_stct:



### Data Fields

- [int flags](#)
- [lbm\\_uint\\_t queue\\_id](#)
- [char queue \[LBM\\_UMQ\\_MAX\\_QUEUE\\_STRLEN\]](#)
- [lbm\\_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 lbm\\_msg\\_umq\\_index\\_assigned\\_ex\\_t\\_stct::flags](#)

Flags that indicate why the index was assigned to the receiver, etc.

##### 7.29.2.2 [lbm\\_uint\\_t lbm\\_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:

- [lbm.h](#)

## 7.30 `lbm_msg_umq_index_assignment_eligibility_start_complete_ex_t_stct` Struct Reference

Structure that holds index assignment information for receivers.

```
#include <lbm.h>
```

### Data Fields

- `int flags`
- `lbm_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 lbm_msg_umq_index_assignment_eligibility_start_complete_ex_t_stct::flags`

Flags that indicate which optional portions are included

##### 7.30.2.2 `lbm_uint_t lbm_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:

- `lbm.h`

## 7.31 `lbm_msg_umq_index_assignment_eligibility_stop_complete_ex_t_stct` Struct Reference

Structure that holds index assignment information for receivers.

```
#include <lbm.h>
```

### Data Fields

- `int flags`
- `lbm_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 lbm_msg_umq_index_assignment_eligibility_stop_complete_ex_t_stct::flags`

Flags that indicate which optional portions are included

##### 7.31.2.2 `lbm_uint_t lbm_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:

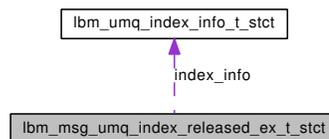
- `lbm.h`

## 7.32 `lbm_msg_umq_index_released_ex_t_stct` Struct Reference

Structure that holds end-of-index information for receivers.

```
#include <lbm.h>
```

Collaboration diagram for `lbm_msg_umq_index_released_ex_t_stct`:



### Data Fields

- `int flags`
- `lbm_uint_t queue_id`
- `char queue [LBM_UMQ_MAX_QUEUE_STRLEN]`
- `lbm_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 lbm_msg_umq_index_released_ex_t_stct::flags`

Flags that indicate why the index was released (user-requested release, etc.)

##### 7.32.2.2 `lbm_uint_t lbm_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:

- `lbm.h`

## 7.33 `lbm_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 <lbm.h>
```

### Data Fields

- `int flags`
- `lbm_uint_t queue_id`
- `lbm_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 `lbm_uint_t lbm_msg_umq_registration_complete_ex_t_stct::assignment_id`

The generated Assignment ID for the receiver with the queue

##### 7.33.2.2 `int lbm_msg_umq_registration_complete_ex_t_stct::flags`

Flags that indicate which optional portions are included

##### 7.33.2.3 `char lbm_msg_umq_registration_complete_ex_t_stct::queue`[LBM\_UMQ\_MAX\_QUEUE\_STRLEN]

The name of the queue registered with

##### 7.33.2.4 `lbm_uint_t lbm_msg_umq_registration_complete_ex_t_stct::queue_id`

The Queue ID of the queue

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.34 lbm\_rcv\_src\_notification\_func\_t\_stct Struct Reference

Structure that holds the application callback for source status notifications for receivers.

```
#include <lbm.h>
```

### Data Fields

- [lbm\\_rcv\\_src\\_notification\\_create\\_function\\_cb](#) **create\_func**
- [lbm\\_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:

- [lbm.h](#)

## 7.35 `lbm_rcv_topic_stats_t_stct` Struct Reference

Structure that holds statistics for a receiver topic.

```
#include <lbm.h>
```

### Data Fields

- char `topic` [LBM\_MSG\_MAX\_TOPIC\_LEN]
- `lbm_uint32_t` `flags`
- char `source` [LBM\_MSG\_MAX\_SOURCE\_LEN]
- `lbm_uint8_t` `otid` [LBM\_OTID\_BLOCK\_SZ]
- `lbm_uint32_t` `topic_idx`

### 7.35.1 Detailed Description

THIS STRUCTURE IS UNSUPPORTED.

### 7.35.2 Field Documentation

#### 7.35.2.1 `lbm_uint32_t lbm_rcv_topic_stats_t_stct::flags`

Flags for the receiver. THIS FIELD IS UNSUPPORTED.

#### 7.35.2.2 `lbm_uint8_t lbm_rcv_topic_stats_t_stct::otid`[LBM\_OTID\_BLOCK\_SZ]

Originating transport ID for a transport joined by the receiver. THIS FIELD IS UNSUPPORTED.

#### 7.35.2.3 `char lbm_rcv_topic_stats_t_stct::source`[LBM\_MSG\_MAX\_SOURCE\_LEN]

Source string for a transport joined by the receiver. THIS FIELD IS UNSUPPORTED.

#### 7.35.2.4 `char lbm_rcv_topic_stats_t_stct::topic`[LBM\_MSG\_MAX\_TOPIC\_LEN]

Topic for the receiver. THIS FIELD IS UNSUPPORTED.

### 7.35.2.5 `lbm_uint32_t lbm_rcv_topic_stats_t_stct::topic_idx`

Topic index for a transport joined by the receiver. THIS FIELD IS UNSUPPORTED.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.36 `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`

#### 7.36.1 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 for backward compatibility only.

#### 7.36.2 Field Documentation

##### 7.36.2.1 `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](#)

## 7.37 `lbm_rcv_transport_stats_lbtipc_t_stct` Struct Reference

Structure that holds datagram statistics for receiver LBT-IPC transports.

```
#include <lbm.h>
```

### Data Fields

- `lbm_ulong_t msgs_rcved`
- `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`

### 7.37.1 Field Documentation

#### 7.37.1.1 `lbm_ulong_t lbm_rcv_transport_stats_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.

#### 7.37.1.2 `lbm_ulong_t lbm_rcv_transport_stats_lbtipc_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.

#### 7.37.1.3 `lbm_ulong_t lbm_rcv_transport_stats_lbtipc_t_stct::lbm_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 `lbm_msgs_no_topic_rcved` counter (below).

**7.37.1.4** `lbm_ulong_t lbm_rcv_transport_stats_lbtipc_t_stct::lbm_reqs_rcved`

Number of UM request messages received (message type LBM\_MSG\_REQUEST).

**7.37.1.5** `lbm_ulong_t lbm_rcv_transport_stats_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).

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.38 `lbm_rcv_transport_stats_lbtrdma_t_stct` Struct Reference

Structure that holds datagram statistics for receiver LBT-RDMA transports.

```
#include <lbm.h>
```

### Data Fields

- `lbm_ulong_t msgs_rcved`
- `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`

### 7.38.1 Field Documentation

#### 7.38.1.1 `lbm_ulong_t lbm_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.

#### 7.38.1.2 `lbm_ulong_t lbm_rcv_transport_stats_lbtrdma_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.

#### 7.38.1.3 `lbm_ulong_t lbm_rcv_transport_stats_lbtrdma_t_stct::lbm_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 `lbm_msgs_no_topic_rcved` counter (below).

**7.38.1.4** `lbm_ulong_t lbm_rcv_transport_stats_lbtrdma_t_stct::lbm_reqs_rcved`

Number of UM request messages received (message type LBM\_MSG\_REQUEST).

**7.38.1.5** `lbm_ulong_t lbm_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).

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.39 lbm\_rcv\_transport\_stats\_lbtrm\_t\_stct Struct Reference

Structure that holds datagram statistics for receiver LBT-RM transports.

```
#include <lbm.h>
```

### Data Fields

- lbm\_ulong\_t [msgs\\_rcved](#)
- lbm\_ulong\_t [bytes\\_rcved](#)
- lbm\_ulong\_t [nak\\_pckts\\_sent](#)
- lbm\_ulong\_t [naks\\_sent](#)
- lbm\_ulong\_t [lost](#)
- lbm\_ulong\_t [ncfs\\_ignored](#)
- lbm\_ulong\_t [ncfs\\_shed](#)
- lbm\_ulong\_t [ncfs\\_rx\\_delay](#)
- lbm\_ulong\_t [ncfs\\_unknown](#)
- lbm\_ulong\_t [nak\\_stm\\_min](#)
- lbm\_ulong\_t [nak\\_stm\\_mean](#)
- lbm\_ulong\_t [nak\\_stm\\_max](#)
- lbm\_ulong\_t [nak\\_tx\\_min](#)
- lbm\_ulong\_t [nak\\_tx\\_mean](#)
- lbm\_ulong\_t [nak\\_tx\\_max](#)
- lbm\_ulong\_t [duplicate\\_data](#)
- lbm\_ulong\_t [unrecovered\\_txw](#)
- lbm\_ulong\_t [unrecovered\\_tmo](#)
- lbm\_ulong\_t [lbm\\_msgs\\_rcved](#)
- lbm\_ulong\_t [lbm\\_msgs\\_no\\_topic\\_rcved](#)
- lbm\_ulong\_t [lbm\\_reqs\\_rcved](#)
- lbm\_ulong\_t [dgrams\\_dropped\\_size](#)
- lbm\_ulong\_t [dgrams\\_dropped\\_type](#)
- lbm\_ulong\_t [dgrams\\_dropped\\_version](#)
- lbm\_ulong\_t [dgrams\\_dropped\\_hdr](#)
- lbm\_ulong\_t [dgrams\\_dropped\\_other](#)
- lbm\_ulong\_t [out\\_of\\_order](#)

### 7.39.1 Field Documentation

#### 7.39.1.1 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.

**7.39.1.2** `lbm_ulong_t lbm_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.

**7.39.1.3** `lbm_ulong_t lbm_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.

**7.39.1.4** `lbm_ulong_t lbm_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.

**7.39.1.5** `lbm_ulong_t lbm_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.

**7.39.1.6** `lbm_ulong_t lbm_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.

**7.39.1.7** `lbm_ulong_t lbm_rcv_transport_stats_lbtrm_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.

### 7.39.1.8 `lbm_ulong_t lbm_rcv_transport_stats_lbtrm_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.

### 7.39.1.9 `lbm_ulong_t lbm_rcv_transport_stats_lbtrm_t_stct::lbm_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 `lbm_msgs_no_topic_rcved` counter (below).

### 7.39.1.10 `lbm_ulong_t lbm_rcv_transport_stats_lbtrm_t_stct::lbm_reqs_rcved`

Number of UM request messages received (message type `LBM_MSG_REQUEST`).

### 7.39.1.11 `lbm_ulong_t lbm_rcv_transport_stats_lbtrm_t_stct::lost`

Number of LBT-RM datagrams detected as lost. When a gap in transport-level sequence numbers is detected, the "lost" statistic increases by the size of the gap. Lost packets are typically recovered a short time later (also see stats "unrecovered\_txw" and "unrecovered\_tmo"). Note that a network might reorder UDP packets. If LBT-RM datagrams arrive out of order, the "lost" statistic is incremented. Subtracting the "out\_of\_order" stat from "lost" shows a more accurate count of lost datagrams.

### 7.39.1.12 `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).

**7.39.1.13** `lbm_ulong_t lbm_rcv_transport_stats_lbtrm_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).

**7.39.1.14** `lbm_ulong_t lbm_rcv_transport_stats_lbtrm_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.

**7.39.1.15** `lbm_ulong_t lbm_rcv_transport_stats_lbtrm_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 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.

**7.39.1.16** `lbm_ulong_t lbm_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.

**7.39.1.17** `lbm_ulong_t lbm_rcv_transport_stats_lbtrm_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.

**7.39.1.18** `lbm_ulong_t lbm_rcv_transport_stats_lbtrm_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.

**7.39.1.19** `lbm_ulong_t lbm_rcv_transport_stats_lbtrm_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.

**7.39.1.20** `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 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_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).

**7.39.1.21** `lbm_ulong_t lbm_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.

**7.39.1.22** `lbm_ulong_t lbm_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.

**7.39.1.23 `lbm_ulong_t lbm_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.

**7.39.1.24 `lbm_ulong_t lbm_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.

**7.39.1.25 `lbm_ulong_t lbm_rcv_transport_stats_lbtrm_t_stct::out_of_order`**

Number of out-of-order LBT-RM datagrams received. A datagram is counted as out of order if it has a sequence number lower than the highest-received so far, but was not tagged as a retransmission.

**7.39.1.26 `lbm_ulong_t lbm_rcv_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.

**7.39.1.27 `lbm_ulong_t lbm_rcv_transport_stats_lbtrm_t_stct::unrecovered_twx`**

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).

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.40 `lbm_rcv_transport_stats_lbtru_t_stct` Struct Reference

Structure that holds datagram statistics for receiver LBT-RU transports.

```
#include <lbm.h>
```

### Data Fields

- `lbm_ulong_t` [msgs\\_rcved](#)
- `lbm_ulong_t` [bytes\\_rcved](#)
- `lbm_ulong_t` [nak\\_pckts\\_sent](#)
- `lbm_ulong_t` [naks\\_sent](#)
- `lbm_ulong_t` [lost](#)
- `lbm_ulong_t` [ncfs\\_ignored](#)
- `lbm_ulong_t` [ncfs\\_shed](#)
- `lbm_ulong_t` [ncfs\\_rx\\_delay](#)
- `lbm_ulong_t` [ncfs\\_unknown](#)
- `lbm_ulong_t` [nak\\_stm\\_min](#)
- `lbm_ulong_t` [nak\\_stm\\_mean](#)
- `lbm_ulong_t` [nak\\_stm\\_max](#)
- `lbm_ulong_t` [nak\\_tx\\_min](#)
- `lbm_ulong_t` [nak\\_tx\\_mean](#)
- `lbm_ulong_t` [nak\\_tx\\_max](#)
- `lbm_ulong_t` [duplicate\\_data](#)
- `lbm_ulong_t` [unrecovered\\_txw](#)
- `lbm_ulong_t` [unrecovered\\_tmo](#)
- `lbm_ulong_t` [lbm\\_msgs\\_rcved](#)
- `lbm_ulong_t` [lbm\\_msgs\\_no\\_topic\\_rcved](#)
- `lbm_ulong_t` [lbm\\_reqs\\_rcved](#)
- `lbm_ulong_t` [dgrams\\_dropped\\_size](#)
- `lbm_ulong_t` [dgrams\\_dropped\\_type](#)
- `lbm_ulong_t` [dgrams\\_dropped\\_version](#)
- `lbm_ulong_t` [dgrams\\_dropped\\_hdr](#)
- `lbm_ulong_t` [dgrams\\_dropped\\_sid](#)
- `lbm_ulong_t` [dgrams\\_dropped\\_other](#)

### 7.40.1 Field Documentation

#### 7.40.1.1 `lbm_ulong_t lbm_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.

**7.40.1.2** `lbm_ulong_t lbm_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.

**7.40.1.3** `lbm_ulong_t lbm_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.

**7.40.1.4** `lbm_ulong_t lbm_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.

**7.40.1.5** `lbm_ulong_t lbm_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.

**7.40.1.6** `lbm_ulong_t lbm_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.

**7.40.1.7** `lbm_ulong_t lbm_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.

**7.40.1.8** `lbm_ulong_t lbm_rcv_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.

#### **7.40.1.9 `lbm_ulong_t lbm_rcv_transport_stats_lbtru_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.

#### **7.40.1.10 `lbm_ulong_t lbm_rcv_transport_stats_lbtru_t_stct::lbm_msgs_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_lbtru_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 `lbm_msgs_no_topic_rcved` counter (below).

#### **7.40.1.11 `lbm_ulong_t lbm_rcv_transport_stats_lbtru_t_stct::lbm_reqs_rcved`**

Number of UM request messages received (message type `LBM_MSG_REQUEST`).

#### **7.40.1.12 `lbm_ulong_t lbm_rcv_transport_stats_lbtru_t_stct::lost`**

Number of LBT-RU datagrams detected as lost.

#### **7.40.1.13 `lbm_ulong_t lbm_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).

#### **7.40.1.14 `lbm_ulong_t lbm_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).

**7.40.1.15** `lbm_ulong_t lbm_rcv_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.

**7.40.1.16** `lbm_ulong_t lbm_rcv_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.

**7.40.1.17** `lbm_ulong_t lbm_rcv_transport_stats_lbtru_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_lbtru_nak_backoff_interval`, it may be taking multiple NAKs to initiate retransmissions, indicating a lossy network.

**7.40.1.18** `lbm_ulong_t lbm_rcv_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.

**7.40.1.19** `lbm_ulong_t lbm_rcv_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.

**7.40.1.20** `lbm_ulong_t lbm_rcv_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.

**7.40.1.21 `lbm_ulong_t lbm_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).

**7.40.1.22 `lbm_ulong_t lbm_rcv_transport_stats_lbtru_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_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.

**7.40.1.23 `lbm_ulong_t lbm_rcv_transport_stats_lbtru_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.

**7.40.1.24 `lbm_ulong_t lbm_rcv_transport_stats_lbtru_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.

**7.40.1.25** `lbm_ulong_t lbm_rcv_transport_stats_lbtru_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_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.

**7.40.1.26** `lbm_ulong_t lbm_rcv_transport_stats_lbtru_t_stct::unrecovered_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.

**7.40.1.27** `lbm_ulong_t lbm_rcv_transport_stats_lbtru_t_stct::unrecovered_twx`

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 `lbtru_transmission_window_size` (default 24MB).

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.41 `lbm_rcv_transport_stats_lbtsmx_t_stct` Struct Reference

Structure that holds datagram statistics for receiver LBT-SMX transports.

```
#include <lbm.h>
```

### Data Fields

- `lbm_ulong_t msgs_rcved`
- `lbm_ulong_t bytes_rcved`
- `lbm_ulong_t lbm_msgs_rcved`
- `lbm_ulong_t lbm_msgs_no_topic_rcved`
- `lbm_ulong_t reserved1`

### 7.41.1 Field Documentation

#### 7.41.1.1 `lbm_ulong_t lbm_rcv_transport_stats_lbtsmx_t_stct::bytes_rcved`

Number of LBT-SMX datagram bytes received, i.e., the total of lengths of all LBT-SMX packets including UM header information.

#### 7.41.1.2 `lbm_ulong_t lbm_rcv_transport_stats_lbtsmx_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.

#### 7.41.1.3 `lbm_ulong_t lbm_rcv_transport_stats_lbtsmx_t_stct::lbm_msgs_rcved`

Number of messages received over an LBT-SMX transport. A single datagram may contain more than more messages provided the messages size is less than half the size of the configuration option `transport_lbtsmx_datagram_max_size` (default 8192 bytes), This number also includes messages received for which there was no interested receiver, tallied in the `lbm_msgs_no_topic_rcved` counter.

#### 7.41.1.4 `lbm_ulong_t lbm_rcv_transport_stats_lbtsmx_t_stct::msgs_rcved`

Number of LBT-SMX datagrams received.

#### 7.41.1.5 `lbm_ulong_t lbm_rcv_transport_stats_lbtsmx_t_stct::reserved1`

Reserved field. Do not use.

The documentation for this struct was generated from the following file:

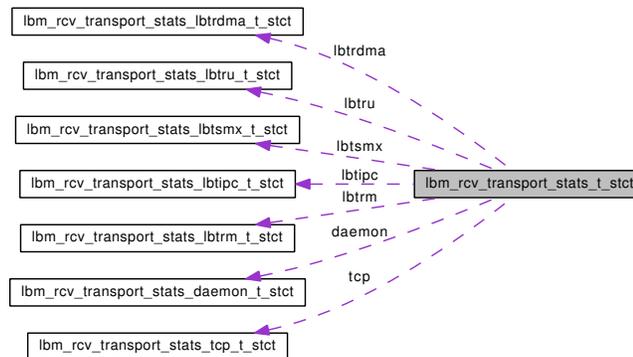
- [lbm.h](#)

## 7.42 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_lbtipc_t lbtipc`
  - `lbm_rcv_transport_stats_lbtrdma_t lbtrdma`
  - `lbm_rcv_transport_stats_lbtsmx_t lbtsmx`
- `} transport`
- `char fill [LBM_EXTERNAL_STRUCT_FILL_SIZE]`

### 7.42.1 Detailed Description

This structure holds statistics for all receiver transports. The structure is filled in when statistics for receiver transports are requested.

## 7.42.2 Field Documentation

### 7.42.2.1 `lbm_rcv_transport_stats_daemon_t` `lbm_rcv_transport_stats_t_stct::daemon`

These statistics have been deprecated.

### 7.42.2.2 `lbm_rcv_transport_stats_lbtipec_t` `lbm_rcv_transport_stats_t_stct::lbtipec`

The statistics for receiver LBT-IPC transports.

### 7.42.2.3 `lbm_rcv_transport_stats_lbtrdma_t` `lbm_rcv_transport_stats_t_stct::lbtrdma`

The statistics for receiver LBT-RDMA transports.

### 7.42.2.4 `lbm_rcv_transport_stats_lbtrm_t` `lbm_rcv_transport_stats_t_stct::lbtrm`

The statistics for receiver LBT-RM transports.

### 7.42.2.5 `lbm_rcv_transport_stats_lbtru_t` `lbm_rcv_transport_stats_t_stct::lbtru`

The statistics for receiver LBT-RU transports.

### 7.42.2.6 `lbm_rcv_transport_stats_lbtsmx_t` `lbm_rcv_transport_stats_t_stct::lbtsmx`

The statistics for receiver LBT-SMX transports.

### 7.42.2.7 `char lbm_rcv_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`.

### 7.42.2.8 `lbm_rcv_transport_stats_tcp_t` `lbm_rcv_transport_stats_t_stct::tcp`

The statistics for receiver TCP transports.

**7.42.2.9** `int lbm_rcv_transport_stats_t_stct::type`

Type of transport (e.g., LBM\_TRANSPORT\_STAT\_TCP, LBM\_TRANSPORT\_STAT\_LBTRM, etc.).

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.43 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](#)

#### 7.43.1 Field Documentation

##### 7.43.1.1 [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.

##### 7.43.1.2 [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.

##### 7.43.1.3 [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).

**7.43.1.4** `lbm_ulong_t lbm_rcv_transport_stats_tcp_t_stct::lbm_reqs_rcved`

Number of UM request messages received (message type LBM\_MSG\_REQUEST).

The documentation for this struct was generated from the following file:

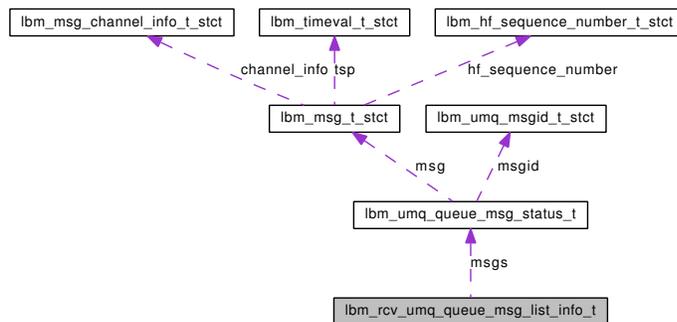
- [lbm.h](#)

## 7.44 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`

#### 7.44.1 Field Documentation

##### 7.44.1.1 `lbm_umq_queue_msg_status_t* lbm_rcv_umq_queue_msg_list_info_t::msgs`

An array of queued messages.

##### 7.44.1.2 `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:

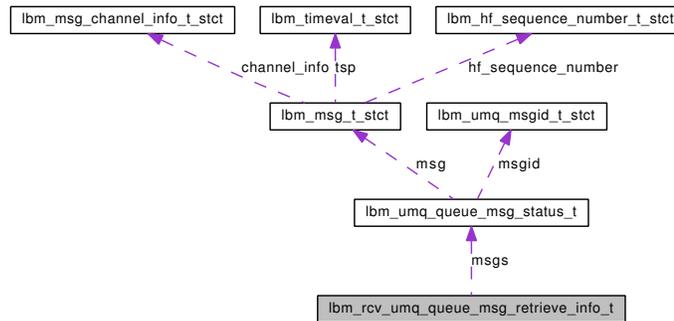
- `lbm.h`

## 7.45 lbm\_rcv\_umq\_queue\_msg\_retrieve\_info\_t Struct Reference

Struct containing an array of UMQ messages retrieved via `lbm_rcv_umq_queue_msg_retrieve`.

```
#include <lbm.h>
```

Collaboration diagram for `lbm_rcv_umq_queue_msg_retrieve_info_t`:



### Data Fields

- `lbm_umq_queue_msg_status_t * msgs`
- `int num_msgs`

### 7.45.1 Field Documentation

#### 7.45.1.1 `lbm_umq_queue_msg_status_t* lbm_rcv_umq_queue_msg_retrieve_info_t::msgs`

An array of the retrieved messages.

#### 7.45.1.2 `int lbm_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:

- `lbm.h`

## 7.46 lbm\_resolver\_event\_advertisement\_t\_stct Struct Reference

Advertisement event structure (for internal use only).

```
#include <lbm.h>
```

### Data Fields

- lbm\_uint32\_t **flags**
- char **topic\_string** [LBM\_MSG\_MAX\_TOPIC\_LEN]
- char **transport\_string** [LBM\_MSG\_MAX\_SOURCE\_LEN]
- lbm\_uint32\_t **topic\_index**
- lbm\_uint32\_t **source\_domain\_id**
- lbm\_uint8\_t **otid** [LBM\_OTID\_BLOCK\_SZ]
- lbm\_uint32\_t **source\_type**

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.47 `lbm_resolver_event_func_t_stct` Struct Reference

Resolver event function (for internal use only).

```
#include <lbm.h>
```

### Data Fields

- [lbm\\_resolver\\_event\\_cb\\_func](#) `event_cb_func`
- `void * clientd`

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.48 lbm\_resolver\_event\_info\_t\_stct Struct Reference

Resolver event structure (for internal use only).

```
#include <lbm.h>
```

### Data Fields

- lbm\_uint64\_t **info\_flags**
- lbm\_uint32\_t **capability\_flags**
- lbm\_uint8\_t **source\_type**
- lbm\_uint32\_t **domain\_id**
- lbm\_uint32\_t **version**
- lbm\_uint8\_t **source\_id** [8]

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.49 `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](#)

## 7.50 `lbm_src_cost_func_t_stct` Struct Reference

Structure that holds the "source\_cost\_evaluation\_function" context attribute.

```
#include <lbm.h>
```

### Data Fields

- [lbm\\_src\\_cost\\_function\\_cb](#) `cost_cb`
- `void * clientd`

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.51 `lbm_src_event_flight_size_notification_t_stct` Struct Reference

Structure that holds flight size notification event data.

```
#include <lbm.h>
```

### Data Fields

- `int type`
- `int state`

#### 7.51.1 Detailed Description

A structure used to indicate a state change in flight size status

#### 7.51.2 Field Documentation

##### 7.51.2.1 `int lbm_src_event_flight_size_notification_t_stct::state`

Current state of specified flight size

##### 7.51.2.2 `int lbm_src_event_flight_size_notification_t_stct::type`

Specifies which flight size's state changed

The documentation for this struct was generated from the following file:

- `lbm.h`

## 7.52 `lbm_src_event_sequence_number_info_t_stct` Struct Reference

Structure that holds sequence number information for a message sent by a source.

```
#include <lbm.h>
```

### Data Fields

- `int flags`
- `lbm_uint_t first_sequence_number`
- `lbm_uint_t last_sequence_number`
- `void * msg_clientd`

#### 7.52.1 Detailed Description

A structure used with UM sources that informs the application the sequence numbers used with a message.

See also:

[lbm\\_src\\_send\\_ex](#)

#### 7.52.2 Field Documentation

##### 7.52.2.1 `lbm_uint_t lbm_src_event_sequence_number_info_t_stct::first_sequence_number`

First sequence number for the message set

##### 7.52.2.2 `int lbm_src_event_sequence_number_info_t_stct::flags`

Flags that indicate which optional portions are included

##### 7.52.2.3 `lbm_uint_t lbm_src_event_sequence_number_info_t_stct::last_sequence_number`

Last sequence number for the message set

**7.52.2.4 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](#)

## 7.53 lbm\_src\_event\_ume\_ack\_ex\_info\_t\_stct Struct Reference

Structure that holds ACK information for a given message in an extended form.

```
#include <lbm.h>
```

### Data Fields

- int [flags](#)
- lbm\_uint\_t [sequence\\_number](#)
- lbm\_uint\_t [rcv\\_registration\\_id](#)
- char [store](#) [LBM\_UME\_MAX\_STORE\_STRLEN]
- void \* [msg\\_clientd](#)
- lbm\_ushort\_t [store\\_index](#)

#### 7.53.1 Detailed Description

A structure used with UMP sources to indicate message acknowledgment by the store and UMP receivers or message stability information (extended form).

#### 7.53.2 Field Documentation

##### 7.53.2.1 int [lbm\\_src\\_event\\_ume\\_ack\\_ex\\_info\\_t\\_stct::flags](#)

Flags

##### 7.53.2.2 void\* [lbm\\_src\\_event\\_ume\\_ack\\_ex\\_info\\_t\\_stct::msg\\_clientd](#)

The clientd pointer passed in for the message

##### 7.53.2.3 lbm\_uint\_t [lbm\\_src\\_event\\_ume\\_ack\\_ex\\_info\\_t\\_stct::rcv\\_registration\\_id](#)

The registration ID for the receiver, if applicable. This field is 0 (zero) if this struct is not being delivered as part of a confirmed delivery event.

##### 7.53.2.4 lbm\_uint\_t [lbm\\_src\\_event\\_ume\\_ack\\_ex\\_info\\_t\\_stct::sequence\\_number](#)

The sequence number of the message

**7.53.2.5** `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 event applies to multiple stores or if using the structure for a delivery confirmation event.

**7.53.2.6** `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 event applies to multiple stores or if using the structure for a delivery confirmation event.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.54 `lbm_src_event_ume_ack_info_t_stct` Struct Reference

Structure that holds ACK information for a given message.

```
#include <lbm.h>
```

### Data Fields

- `lbm_uint_t` [sequence\\_number](#)
- `lbm_uint_t` [rcv\\_registration\\_id](#)
- `void *` [msg\\_clientd](#)

#### 7.54.1 Detailed Description

A structure used with UMP sources to indicate message acknowledgment by the store and UMP receivers.

#### 7.54.2 Field Documentation

##### 7.54.2.1 `void* lbm_src_event_ume_ack_info_t_stct::msg_clientd`

The `clientd` pointer passed in for the message

##### 7.54.2.2 `lbm_uint_t lbm_src_event_ume_ack_info_t_stct::rcv_registration_id`

The registration ID for the receiver

##### 7.54.2.3 `lbm_uint_t lbm_src_event_ume_ack_info_t_stct::sequence_number`

The sequence number of the message that is being acknowledged

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.55 `lbm_src_event_ume_deregistration_ex_t_stct` Struct Reference

Structure that holds store deregistration information for the UMP source in an extended form.

```
#include <lbm.h>
```

### Data Fields

- `int flags`
- `lbm_uint32_t registration_id`
- `lbm_uint_t sequence_number`
- `lbm_ushort_t store_index`
- `char store` [LBM\_UME\_MAX\_STORE\_STRLEN]

### 7.55.1 Detailed Description

A structure used with UMP sources to indicate successful deregistration (extended form).

### 7.55.2 Field Documentation

#### 7.55.2.1 `int lbm_src_event_ume_deregistration_ex_t_stct::flags`

Flags

#### 7.55.2.2 `lbm_uint32_t lbm_src_event_ume_deregistration_ex_t_stct::registration_id`

The registration ID for the source

#### 7.55.2.3 `lbm_uint_t lbm_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

#### 7.55.2.4 `char lbm_src_event_ume_deregistration_ex_t_stct::store[LBM_UME_MAX_STORE_STRLEN]`

The store that was registered with.

#### 7.55.2.5 `lbm_ushort_t lbm_src_event_ume_deregistration_ex_t_stct::store_index`

The store index of the store involved.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.56 `lbm_src_event_ume_registration_complete_ext_stct` Struct Reference

Structure that holds information for sources after registration is complete to all involved stores.

```
#include <lbm.h>
```

### Data Fields

- `int flags`
- `lbm_uint_t sequence_number`

#### 7.56.1 Detailed Description

A structure used with UMP sources to indicate successful registration to quorum or to all stores involved.

#### 7.56.2 Field Documentation

##### 7.56.2.1 `int lbm_src_event_ume_registration_complete_ext_stct::flags`

Flags

##### 7.56.2.2 `lbm_uint_t lbm_src_event_ume_registration_complete_ext_stct::sequence_number`

The sequence number that will be the first sent

The documentation for this struct was generated from the following file:

- `lbm.h`

## 7.57 `lbm_src_event_ume_registration_ex_t_stct` Struct Reference

Structure that holds store registration information for the UMP source in an extended form.

```
#include <lbm.h>
```

### Data Fields

- `int flags`
- `lbm_uint_t registration_id`
- `lbm_uint_t sequence_number`
- `lbm_ushort_t store_index`
- `char store` [LBM\_UME\_MAX\_STORE\_STRLEN]

### 7.57.1 Detailed Description

A structure used with UMP sources to indicate successful registration (extended form).

### 7.57.2 Field Documentation

#### 7.57.2.1 `int lbm_src_event_ume_registration_ex_t_stct::flags`

Flags

#### 7.57.2.2 `lbm_uint_t lbm_src_event_ume_registration_ex_t_stct::registration_id`

The registration ID for the source

#### 7.57.2.3 `lbm_uint_t lbm_src_event_ume_registration_ex_t_stct::sequence_number`

The sequence number for the source to use as reported by the store

#### 7.57.2.4 `char lbm_src_event_ume_registration_ex_t_stct::store`[LBM\_UME\_MAX\_STORE\_STRLEN]

The store that was registered with

**7.57.2.5** `lbm_ushort_t lbm_src_event_ume_registration_ex_t_stct::store_index`

The store index of the store involved

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.58 `lbm_src_event_ume_registration_t_stct` Struct Reference

Structure that holds store registration information for the UMP source.

```
#include <lbm.h>
```

### Data Fields

- `lbm_uint_t registration_id`

#### 7.58.1 Detailed Description

A structure used with UMP sources to indicate successful registration.

#### 7.58.2 Field Documentation

##### 7.58.2.1 `lbm_uint_t lbm_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:

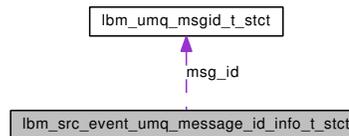
- `lbm.h`

## 7.59 lbm\_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 <lbm.h>
```

Collaboration diagram for lbm\_src\_event\_umq\_message\_id\_info\_t\_stct:



### Data Fields

- int [flags](#)
- [lbm\\_umq\\_msgid\\_t](#) msg\_id
- void \* [msg\\_clientd](#)

### 7.59.1 Detailed Description

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.

### 7.59.2 Field Documentation

#### 7.59.2.1 int [lbm\\_src\\_event\\_umq\\_message\\_id\\_info\\_t\\_stct::flags](#)

Flags that indicate which optional portions are included

#### 7.59.2.2 void\* [lbm\\_src\\_event\\_umq\\_message\\_id\\_info\\_t\\_stct::msg\\_clientd](#)

The clientd pointer passed in for the message

#### 7.59.2.3 [lbm\\_umq\\_msgid\\_t](#) [lbm\\_src\\_event\\_umq\\_message\\_id\\_info\\_t\\_stct::msg\\_id](#)

Message ID for the message sent

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.60 `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]`

### 7.60.1 Detailed Description

A structure used with UMQ sources to indicate successful source registration to quorum or to all queue instances involved.

### 7.60.2 Field Documentation

#### 7.60.2.1 `int lbm_src_event_umq_registration_complete_ex_t_stct::flags`

Flags that indicate which optional portions are included

#### 7.60.2.2 `char lbm_src_event_umq_registration_complete_ex_t_stct::queue[LBM_UMQ_MAX_QUEUE_STRLEN]`

The name of the queue registered with

#### 7.60.2.3 `lbm_uint_t lbm_src_event_umq_registration_complete_ex_t_stct::queue_id`

The Queue ID of the queue

The documentation for this struct was generated from the following file:

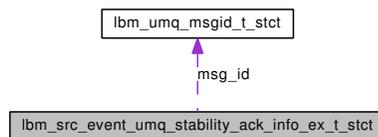
- `lbm.h`

## 7.61 lbm\_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 <lbm.h>
```

Collaboration diagram for lbm\_src\_event\_umq\_stability\_ack\_info\_ex\_t\_stct:



### Data Fields

- int `flags`
- `lbm_umq_msgid_t` `msg_id`
- `lbm_uint_t` `first_sequence_number`
- `lbm_uint_t` `last_sequence_number`
- `lbm_uint_t` `queue_id`
- `lbm_uint_t` `queue_instance_index`
- void \* `msg_clientd`
- char `queue_instance` [LBM\_UME\_MAX\_STORE\_STRLEN]
- char `queue` [LBM\_UMQ\_MAX\_QUEUE\_STRLEN]

#### 7.61.1 Detailed Description

A structure used with UMQ source applications to indicate message acknowledgment by a queue instance.

#### 7.61.2 Field Documentation

##### 7.61.2.1 `lbm_uint_t` `lbm_src_event_umq_stability_ack_info_ex_t_stct::first_sequence_number`

First sequence number for the message after being fragmented

##### 7.61.2.2 `int` `lbm_src_event_umq_stability_ack_info_ex_t_stct::flags`

Flags that indicate which optional portions are included

**7.61.2.3** `lbm_uint_t lbm_src_event_umq_stability_ack_info_ex_t_stct::last_sequence_number`

Last sequence number for the message after being fragmented

**7.61.2.4** `void* lbm_src_event_umq_stability_ack_info_ex_t_stct::msg_clientd`

The clientd pointer passed in for the message

**7.61.2.5** `lbm_umq_msgid_t lbm_src_event_umq_stability_ack_info_ex_t_stct::msg_id`

Message ID of the message being acknowledged

**7.61.2.6** `char lbm_src_event_umq_stability_ack_info_ex_t_stct::queue[LBM_UMQ_MAX_QUEUE_STRLEN]`

The name of the queue

**7.61.2.7** `lbm_uint_t lbm_src_event_umq_stability_ack_info_ex_t_stct::queue_id`

The Queue ID of the queue

**7.61.2.8** `char lbm_src_event_umq_stability_ack_info_ex_t_stct::queue_instance[LBM_UME_MAX_STORE_STRLEN]`

The instance of the queue acknowledging the message

**7.61.2.9** `lbm_uint_t lbm_src_event_umq_stability_ack_info_ex_t_stct::queue_instance_index`

The index of the instance of the queue acknowledging the message

The documentation for this struct was generated from the following file:

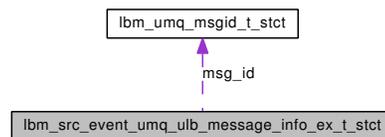
- [lbm.h](#)

## 7.62 lbm\_src\_event\_umq\_ulb\_message\_info\_ex\_t\_stct Struct Reference

Structure that holds UMQ ULB message information in an extended form.

```
#include <lbm.h>
```

Collaboration diagram for lbm\_src\_event\_umq\_ulb\_message\_info\_ex\_t\_stct:



### Data Fields

- int `flags`
- `lbm_umq_msgid_t` `msg_id`
- `lbm_umq_regid_t` `registration_id`
- `lbm_uint_t` `first_sequence_number`
- `lbm_uint_t` `last_sequence_number`
- `lbm_uint_t` `assignment_id`
- `lbm_uint_t` `application_set_index`
- void \* `msg_clientd`
- char `receiver` [LBM\_UMQ\_ULB\_MAX\_RECEIVER\_STRLEN]

#### 7.62.1 Detailed Description

A structure used with UMQ ULB source applications to indicate message events.

#### 7.62.2 Field Documentation

##### 7.62.2.1 `lbm_uint_t lbm_src_event_umq_ulb_message_info_ex_t_stct::application_set_index`

The Application Set Index the receiver is in

##### 7.62.2.2 `lbm_uint_t lbm_src_event_umq_ulb_message_info_ex_t_stct::assignment_id`

The Assignment ID of the receiver

**7.62.2.3** `lbm_uint_t lbm_src_event_umq_ulb_message_info_ex_t_stct::first_sequence_number`

First sequence number for the message after being fragmented

**7.62.2.4** `int lbm_src_event_umq_ulb_message_info_ex_t_stct::flags`

Flags that indicate which optional portions are included

**7.62.2.5** `lbm_uint_t lbm_src_event_umq_ulb_message_info_ex_t_stct::last_sequence_number`

Last sequence number for the message after being fragmented

**7.62.2.6** `void* lbm_src_event_umq_ulb_message_info_ex_t_stct::msg_clientd`

The clientd pointer passed in for the message

**7.62.2.7** `lbm_umq_msgid_t lbm_src_event_umq_ulb_message_info_ex_t_stct::msg_id`

Message ID of the message

**7.62.2.8** `char lbm_src_event_umq_ulb_message_info_ex_t_stct::receiver[LBM_UMQ_ULB_MAX_RECEIVER_STRLEN]`

The receivers immediate message target string

**7.62.2.9** `lbm_umq_regid_t lbm_src_event_umq_ulb_message_info_ex_t_stct::registration_id`

The registration ID of the receiver

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.63 `lbm_src_event_umq_ulb_receiver_info_ex_t_stct` Struct Reference

Structure that holds UMQ ULB receiver information in an extended form.

```
#include <lbm.h>
```

### Data Fields

- `int flags`
- `lbm_umq_regid_t registration_id`
- `lbm_uint_t assignment_id`
- `lbm_uint_t application_set_index`
- `char receiver` [LBM\_UMQ\_ULB\_MAX\_RECEIVER\_STRLEN]

#### 7.63.1 Detailed Description

A structure used with UMQ ULB source applications to indicate receiver events.

#### 7.63.2 Field Documentation

##### 7.63.2.1 `lbm_uint_t lbm_src_event_umq_ulb_receiver_info_ex_t_stct::application_set_index`

The Application Set Index the receiver is in

##### 7.63.2.2 `lbm_uint_t lbm_src_event_umq_ulb_receiver_info_ex_t_stct::assignment_id`

The Assignment ID of the receiver

##### 7.63.2.3 `int lbm_src_event_umq_ulb_receiver_info_ex_t_stct::flags`

Flags that indicate which optional portions are included

##### 7.63.2.4 `char lbm_src_event_umq_ulb_receiver_info_ex_t_stct::receiver`[LBM\_UMQ\_ULB\_MAX\_RECEIVER\_STRLEN]

The receivers immediate message target string

### 7.63.2.5 [lbm\\_umq\\_regid\\_t](#) [lbm\\_src\\_event\\_umq\\_ulb\\_receiver\\_info\\_ext\\_stct::registration\\_id](#)

The registration ID of the receiver

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.64 `lbm_src_event_wakeup_t_stct` Struct Reference

Structure that holds source wakeup event data.

```
#include <lbm.h>
```

### Data Fields

- `int flags`

#### 7.64.1 Detailed Description

A structure used to indicate the type of source that is now unblocked.

#### 7.64.2 Field Documentation

##### 7.64.2.1 `int lbm_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 `LBM_SRC_EVENT_WAKEUP_FLAG_*`

The documentation for this struct was generated from the following file:

- `lbm.h`

## 7.65 `lbm_src_notify_func_t_stct` Struct Reference

Structure that holds the callback for source notifications.

```
#include <lbm.h>
```

### Data Fields

- [lbm\\_src\\_notify\\_function\\_cb](#) **notifyfunc**
- void \* **clientd**

#### 7.65.1 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:

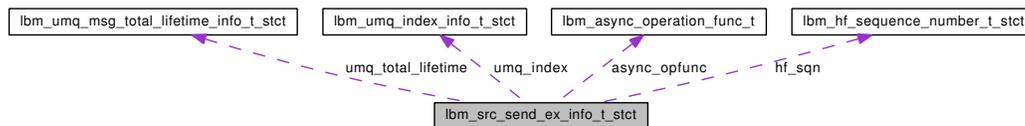
- [lbm.h](#)

## 7.66 lbm\_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 <lbm.h>
```

Collaboration diagram for lbm\_src\_send\_ex\_info\_t\_stct:



### Data Fields

- int [flags](#)
- void \* [ume\\_msg\\_clientd](#)
- lbm\_src\_channel\_info\_t \* [channel\\_info](#)
- lbm\_apphdr\_chain\_t \* [apphdr\\_chain](#)
- lbm\_umq\_index\_info\_t \* [umq\\_index](#)
- lbm\_umq\_msg\_total\_lifetime\_info\_t \* [umq\\_total\\_lifetime](#)
- lbm\_msg\_properties\_t \* [properties](#)
- lbm\_hf\_sequence\_number\_t [hf\\_sqn](#)
- lbm\_async\_operation\_func\_t \* [async\\_opfunc](#)

#### 7.66.1 Detailed Description

See also:

[lbm\\_src\\_send\\_ex](#)

#### 7.66.2 Field Documentation

##### 7.66.2.1 lbm\_apphdr\_chain\_t\* lbm\_src\_send\_ex\_info\_t\_stct::apphdr\_chain

pointer to information used to send messages using app header chains

##### 7.66.2.2 lbm\_async\_operation\_func\_t\* lbm\_src\_send\_ex\_info\_t\_stct::async\_opfunc

pointer to an asynchronous operation callback

**7.66.2.3** [lbm\\_src\\_channel\\_info\\_t\\*](#) [lbm\\_src\\_send\\_ex\\_info\\_t\\_stct::channel\\_info](#)

pointer to information used to send messages on a channel

**7.66.2.4** `int` [lbm\\_src\\_send\\_ex\\_info\\_t\\_stct::flags](#)

Flags that set which settings are active as defined by "LBM\_SRC\_SEND\_EX\_\*". Search [lbm.h](#) for LBM\_SRC\_SEND\_EX\_FLAG\_\*.

**7.66.2.5** [lbm\\_hf\\_sequence\\_number\\_t](#) [lbm\\_src\\_send\\_ex\\_info\\_t\\_stct::hf\\_sqn](#)

The hot failover sequence number to send

**7.66.2.6** [lbm\\_msg\\_properties\\_t\\*](#) [lbm\\_src\\_send\\_ex\\_info\\_t\\_stct::properties](#)

pointer to a message properties structure

**7.66.2.7** `void*` [lbm\\_src\\_send\\_ex\\_info\\_t\\_stct::ume\\_msg\\_clientd](#)

client data pointer to be passed back in source events for stability and confirmations

**7.66.2.8** [lbm\\_umq\\_index\\_info\\_t\\*](#) [lbm\\_src\\_send\\_ex\\_info\\_t\\_stct::umq\\_index](#)

pointer to information used to send messages and associate them with a given UMQ index

**7.66.2.9** [lbm\\_umq\\_msg\\_total\\_lifetime\\_info\\_t\\*](#) [lbm\\_src\\_send\\_ex\\_info\\_t\\_stct::umq\\_total\\_lifetime](#)

pointer to information used to specify a message's total lifetime

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.67 `lbm_src_transport_stats_daemon_t_stct` Struct Reference

Structure that holds statistics for source daemon mode transport (deprecated).

```
#include <lbm.h>
```

### Data Fields

- `lbm_ulong_t bytes_buffered`

#### 7.67.1 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.

#### 7.67.2 Field Documentation

##### 7.67.2.1 `lbm_ulong_t lbm_src_transport_stats_daemon_t_stct::bytes_buffered`

This statistic has been deprecated.

The documentation for this struct was generated from the following file:

- `lbm.h`

## 7.68 `lbm_src_transport_stats_lbtipc_t_stct` Struct Reference

Structure that holds datagram statistics for source LBT-IPC transports.

```
#include <lbm.h>
```

### Data Fields

- `lbm_ulong_t num_clients`
- `lbm_ulong_t msgs_sent`
- `lbm_ulong_t bytes_sent`

### 7.68.1 Field Documentation

#### 7.68.1.1 `lbm_ulong_t lbm_src_transport_stats_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.

#### 7.68.1.2 `lbm_ulong_t lbm_src_transport_stats_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).

#### 7.68.1.3 `lbm_ulong_t lbm_src_transport_stats_lbtipc_t_stct::num_clients`

Number of receiver transports that are currently connected to this source transport.

The documentation for this struct was generated from the following file:

- `lbm.h`

## 7.69 `lbm_src_transport_stats_lbtrdma_t_stct` Struct Reference

Structure that holds datagram statistics for source LBT-RDMA transports.

```
#include <lbm.h>
```

### Data Fields

- `lbm_ulong_t num_clients`
- `lbm_ulong_t msgs_sent`
- `lbm_ulong_t bytes_sent`

#### 7.69.1 Field Documentation

##### 7.69.1.1 `lbm_ulong_t lbm_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.

##### 7.69.1.2 `lbm_ulong_t lbm_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).

##### 7.69.1.3 `lbm_ulong_t lbm_src_transport_stats_lbtrdma_t_stct::num_clients`

Number of receiver transports that are currently connected to this source transport.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.70 `lbm_src_transport_stats_lbtrm_t_stct` Struct Reference

Structure that holds datagram statistics for source LBT-RM transports.

```
#include <lbm.h>
```

### Data Fields

- `lbm_ulong_t` [msgs\\_sent](#)
- `lbm_ulong_t` [bytes\\_sent](#)
- `lbm_ulong_t` [txw\\_msgs](#)
- `lbm_ulong_t` [txw\\_bytes](#)
- `lbm_ulong_t` [nak\\_pkts\\_rcved](#)
- `lbm_ulong_t` [naks\\_rcved](#)
- `lbm_ulong_t` [naks\\_ignored](#)
- `lbm_ulong_t` [naks\\_shed](#)
- `lbm_ulong_t` [naks\\_rx\\_delay\\_ignored](#)
- `lbm_ulong_t` [rxs\\_sent](#)
- `lbm_ulong_t` [rctlr\\_data\\_msgs](#)
- `lbm_ulong_t` [rctlr\\_rx\\_msgs](#)
- `lbm_ulong_t` [rx\\_bytes\\_sent](#)

### 7.70.1 Field Documentation

#### 7.70.1.1 `lbm_ulong_t lbm_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.

#### 7.70.1.2 `lbm_ulong_t lbm_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).

#### 7.70.1.3 `lbm_ulong_t lbm_src_transport_stats_lbtrm_t_stct::nak_pkts_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).

#### 7.70.1.4 `lbm_ulong_t lbm_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.

#### 7.70.1.5 `lbm_ulong_t lbm_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.

#### 7.70.1.6 `lbm_ulong_t lbm_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 NCF `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.

#### 7.70.1.7 `lbm_ulong_t lbm_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.

#### 7.70.1.8 `lbm_ulong_t lbm_src_transport_stats_lbtrm_t_stct::rctrl_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.

**7.70.1.9 `lbm_ulong_t lbm_src_transport_stats_lbtrm_t_stct::retlr_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.

**7.70.1.10 `lbm_ulong_t lbm_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 become 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`).

**7.70.1.11 `lbm_ulong_t lbm_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 become 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`).

**7.70.1.12 `lbm_ulong_t lbm_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.

**7.70.1.13 `lbm_ulong_t lbm_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.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.71 `lbm_src_transport_stats_lbtru_t_stct` Struct Reference

Structure that holds datagram statistics for source LBT-RU transports.

```
#include <lbm.h>
```

### Data Fields

- `lbm_ulong_t msgs_sent`
- `lbm_ulong_t bytes_sent`
- `lbm_ulong_t nak_pckts_rcved`
- `lbm_ulong_t naks_rcved`
- `lbm_ulong_t naks_ignored`
- `lbm_ulong_t naks_shed`
- `lbm_ulong_t naks_rx_delay_ignored`
- `lbm_ulong_t rxs_sent`
- `lbm_ulong_t num_clients`
- `lbm_ulong_t rx_bytes_sent`

### 7.71.1 Field Documentation

#### 7.71.1.1 `lbm_ulong_t lbm_src_transport_stats_lbtru_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.

#### 7.71.1.2 `lbm_ulong_t lbm_src_transport_stats_lbtru_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_lbtru_datagram_max_size` (default 8KB).

#### 7.71.1.3 `lbm_ulong_t lbm_src_transport_stats_lbtru_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).

#### 7.71.1.4 lbm\_ulong\_t lbm\_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.

#### 7.71.1.5 lbm\_ulong\_t lbm\_src\_transport\_stats\_lbtru\_t\_stct::naks\_reved

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.

#### 7.71.1.6 lbm\_ulong\_t lbm\_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 NCF 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.

#### 7.71.1.7 lbm\_ulong\_t lbm\_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.

#### 7.71.1.8 lbm\_ulong\_t lbm\_src\_transport\_stats\_lbtru\_t\_stct::num\_clients

Number of receiver transports that are currently connected to this source transport.

**7.71.1.9** `lbm_ulong_t lbm_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`).

**7.71.1.10** `lbm_ulong_t lbm_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`).

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.72 `lbm_src_transport_stats_lbtsmx_t_stct` Struct Reference

Structure that holds datagram statistics for source LBT-SMX transports.

```
#include <lbm.h>
```

### Data Fields

- `lbm_ulong_t num_clients`
- `lbm_ulong_t msgs_sent`
- `lbm_ulong_t bytes_sent`

#### 7.72.1 Field Documentation

##### 7.72.1.1 `lbm_ulong_t lbm_src_transport_stats_lbtsmx_t_stct::bytes_sent`

Number of LBT-SMX datagram bytes sent, i.e., the total of lengths of all LBT-SMX packets including UM header information.

##### 7.72.1.2 `lbm_ulong_t lbm_src_transport_stats_lbtsmx_t_stct::msgs_sent`

Number of LBT-SMX datagrams sent.

##### 7.72.1.3 `lbm_ulong_t lbm_src_transport_stats_lbtsmx_t_stct::num_clients`

Number of receiver transports that are currently connected to this source transport.

The documentation for this struct was generated from the following file:

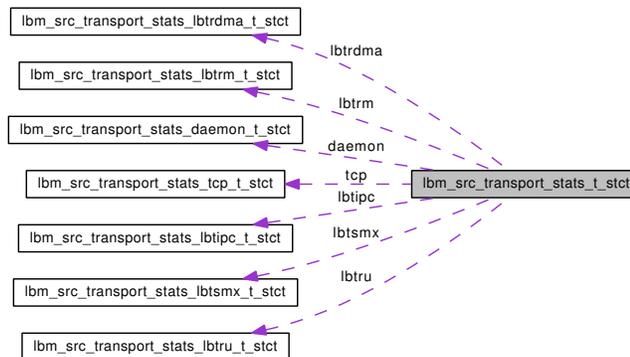
- [lbm.h](#)

## 7.73 lbm\_src\_transport\_stats\_t\_stct Struct Reference

Structure that holds statistics for source transports.

```
#include <lbm.h>
```

Collaboration diagram for lbm\_src\_transport\_stats\_t\_stct:



### Data Fields

- int `type`
- char `source` [LBM\_MSG\_MAX\_SOURCE\_LEN]
- union {
  - `lbm_src_transport_stats_tcp_t tcp`
  - `lbm_src_transport_stats_lbtrm_t lbtrm`
  - `lbm_src_transport_stats_daemon_t daemon`
  - `lbm_src_transport_stats_lbtru_t lbtru`
  - `lbm_src_transport_stats_lbtipc_t lbtipc`
  - `lbm_src_transport_stats_lbtsmx_t lbtsmx`
  - `lbm_src_transport_stats_lbtrdma_t lbtrdma`
- char `_fill` [LBM\_EXTERNAL\_STRUCT\_FILL\_SIZE]

#### 7.73.1 Detailed Description

This structure holds statistics for all source transports. The structure is filled in when statistics for source transports are requested.

## 7.73.2 Field Documentation

### 7.73.2.1 `lbm_src_transport_stats_daemon_t` `lbm_src_transport_stats_t_stct::daemon`

These statistics have been deprecated.

### 7.73.2.2 `lbm_src_transport_stats_lbtipec_t` `lbm_src_transport_stats_t_stct::lbtipec`

The statistics for source LBT-IPC transports.

### 7.73.2.3 `lbm_src_transport_stats_lbtrdma_t` `lbm_src_transport_stats_t_stct::lbtrdma`

The statistics for source LBT-RDMA transports.

### 7.73.2.4 `lbm_src_transport_stats_lbtrm_t` `lbm_src_transport_stats_t_stct::lbtrm`

The statistics for source LBT-RM transports.

### 7.73.2.5 `lbm_src_transport_stats_lbtru_t` `lbm_src_transport_stats_t_stct::lbtru`

The statistics for source LBT-RU transports.

### 7.73.2.6 `lbm_src_transport_stats_lbtstmx_t` `lbm_src_transport_stats_t_stct::lbtstmx`

The statistics for source LBT-SMX transports.

### 7.73.2.7 `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`.

### 7.73.2.8 `lbm_src_transport_stats_tcp_t` `lbm_src_transport_stats_t_stct::tcp`

The statistics for source TCP transports.

**7.73.2.9** `int lbm_src_transport_stats_t_stct::type`

Type of transport (LBM\_TRANSPORT\_STAT\_TCP, LBM\_TRANSPORT\_STAT\_LBTRM, etc.).

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.74 `lbm_src_transport_stats_tcp_t_stct` Struct Reference

Structure that holds datagram statistics for source TCP transports.

```
#include <lbm.h>
```

### Data Fields

- `lbm_ulong_t num_clients`
- `lbm_ulong_t bytes_buffered`

#### 7.74.1 Field Documentation

##### 7.74.1.1 `lbm_ulong_t lbm_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_receiver_behavior` and `transport_session_maximum_buffer`.

##### 7.74.1.2 `lbm_ulong_t lbm_src_transport_stats_tcp_t_stct::num_clients`

Number of TCP receiver clients currently connected over this transport.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.75 `lbm_str_hash_func_ex_t_stct` Struct Reference

Structure that holds the hash function callback information.

```
#include <lbm.h>
```

### Data Fields

- [lbm\\_str\\_hash\\_function\\_cb\\_ex](#) hashfunc
- void \* `clientd`

### 7.75.1 Detailed Description

A structure used with options to set/get a specific hash function information.

### 7.75.2 Field Documentation

#### 7.75.2.1 [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](#)

## 7.76 lbm\_timeval\_t\_stct Struct Reference

Structure that holds seconds and microseconds since midnight, Jan 1, 1970 UTC.

```
#include <lbm.h>
```

### Data Fields

- lbm\_ulong\_t **tv\_sec**
- lbm\_ulong\_t **tv\_usec**

#### 7.76.1 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:

- [lbm.h](#)

## 7.77 `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`
- `lbm_uint32_t transport_idx`
- `char _fill [LBM_EXTERNAL_STRUCT_FILL_SIZE]`

### 7.77.1 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:session\_id[topic\_idx] (session\_id optional, per configuration option `transport_tcp_use_session_id`)  
example: TCP:192.168.0.4:45789:f1789bcc [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](#)

## 7.77.2 Field Documentation

### 7.77.2.1 `lbm_ushort_t lbm_transport_source_info_t_stct::dest_port`

Destination port. Applicable only to LBT-RM. Stored in host order.

### 7.77.2.2 `lbm_uint32_t lbm_transport_source_info_t_stct::mc_group`

Multicast group. Applicable only to LBT-RM. Stored in network order.

### 7.77.2.3 `lbm_uint32_t lbm_transport_source_info_t_stct::session_id`

Session ID. Applicable to all transport. Stored in host order.

### 7.77.2.4 `lbm_uint32_t lbm_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.

### 7.77.2.5 `lbm_ushort_t lbm_transport_source_info_t_stct::src_port`

Source port. Applicable only to LBT-RM, LBT-RU, TCP, and LBT-RDMA. Stored in host order.

### 7.77.2.6 `lbm_uint32_t lbm_transport_source_info_t_stct::topic_idx`

Topic index. Applicable to all transports. Stored in host order.

### 7.77.2.7 `lbm_uint32_t lbm_transport_source_info_t_stct::transport_id`

Transport ID. Applicable only to LBT-IPC. Stored in host order.

**7.77.2.8** `lbm_uint32_t lbm_transport_source_info_t_stct::transport_idx`

Transport index. Applicable to all transports. Stored in host order.

**7.77.2.9** `int lbm_transport_source_info_t_stct::type`

Type of transport. See LBM\_TRANSPORT\_TYPE\_\*.

The documentation for this struct was generated from the following file:

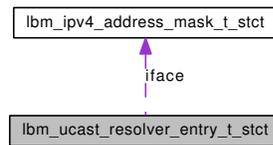
- [lbm.h](#)

## 7.78 lbm\_ucast\_resolver\_entry\_t\_stct Struct Reference

Structure that holds information for a unicast resolver daemon for configuration purposes.

```
#include <lbm.h>
```

Collaboration diagram for lbm\_ucast\_resolver\_entry\_t\_stct:



### Data Fields

- [lbm\\_ipv4\\_address\\_mask\\_t iface](#)
- [lbm\\_uint\\_t resolver\\_ip](#)
- [lbm\\_ushort\\_t source\\_port](#)
- [lbm\\_ushort\\_t destination\\_port](#)

#### 7.78.1 Detailed Description

A structure used with options to get/set information about unicast resolver daemons.

#### 7.78.2 Field Documentation

##### 7.78.2.1 lbm\_ushort\_t lbm\_ucast\_resolver\_entry\_t\_stct::destination\_port

The port configured on the unicast resolver daemon. (network order)

##### 7.78.2.2 lbm\_ipv4\_address\_mask\_t lbm\_ucast\_resolver\_entry\_t\_stct::iface

Interface to be used

##### 7.78.2.3 lbm\_uint\_t lbm\_ucast\_resolver\_entry\_t\_stct::resolver\_ip

The IP address of the unicast resolver daemon (network order)

**7.78.2.4** `lbm_ushort_t lbm_ucast_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)

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.79 `lbm_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 <lbm.h>
```

### Data Fields

- `lbm_ume_ctx_rcv_ctx_notification_create_function_cb` `create_func`
- `lbm_ume_ctx_rcv_ctx_notification_delete_function_cb` `delete_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:

- [lbm.h](#)

## 7.80 `lbm_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 <lbm.h>
```

### Data Fields

- `int flags`
- `lbm_uint_t low_sequence_number`
- `lbm_uint_t low_rxreq_max_sequence_number`
- `lbm_uint_t high_sequence_number`
- `void * source_clientd`
- `char source [LBM_MSG_MAX_SOURCE_LEN]`
- `lbm_uint64_t src_session_id`

### 7.80.1 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:

[lbm\\_ume\\_rcv\\_recovery\\_info\\_ex\\_func\\_t](#)

### 7.80.2 Field Documentation

#### 7.80.2.1 `int lbm_ume_rcv_recovery_info_ex_func_info_t_stct::flags`

Flags that indicate optional portions that are included

#### 7.80.2.2 `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

**7.80.2.3** `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)

**7.80.2.4** `lbm_uint_t lbm_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

**7.80.2.5** `char lbm_ume_rcv_recovery_info_ex_func_info_t_stct::source[LBM_MSG_MAX_SOURCE_LEN]`

The source

**7.80.2.6** `void* lbm_ume_rcv_recovery_info_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

**7.80.2.7** `lbm_uint64_t lbm_ume_rcv_recovery_info_ex_func_info_t_stct::src_session_id`

The session ID for the source

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.81 `lbm_ume_rcv_recovery_info_ex_func_t_stct` Struct Reference

Structure that holds the application callback for recovery sequence number information, extended form.

```
#include <lbm.h>
```

### Data Fields

- [lbm\\_ume\\_rcv\\_recovery\\_info\\_ex\\_function\\_cb](#) **func**
- void \* **clientd**

### 7.81.1 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:

- [lbm.h](#)

## 7.82 lbm\_ume\_rcv\_regid\_ex\_func\_info\_t\_stct Struct Reference

Structure that holds information for UMP receiver registration ID application callbacks.

```
#include <lbm.h>
```

### Data Fields

- int [flags](#)
- lbm\_uint\_t [src\\_registration\\_id](#)
- lbm\_ushort\_t [store\\_index](#)
- void \* [source\\_clientd](#)
- char [source](#) [LBM\_MSG\_MAX\_SOURCE\_LEN]
- char [store](#) [LBM\_UME\_MAX\_STORE\_STRLEN]

### 7.82.1 Detailed Description

A structure used with UMP receiver registration ID callbacks to pass in information.

See also:

[lbm\\_ume\\_rcv\\_regid\\_func\\_t](#)

### 7.82.2 Field Documentation

#### 7.82.2.1 int [lbm\\_ume\\_rcv\\_regid\\_ex\\_func\\_info\\_t\\_stct::flags](#)

Flags that indicate which optional portions are included

#### 7.82.2.2 char [lbm\\_ume\\_rcv\\_regid\\_ex\\_func\\_info\\_t\\_stct::source](#)[LBM\_MSG\_MAX\_SOURCE\_LEN]

The source for the registration ID

#### 7.82.2.3 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

**7.82.2.4** `lbm_uint_t lbm_ume_rcv_regid_ex_func_info_t_stct::src_registration_id`

The registration ID for the source

**7.82.2.5** `char lbm_ume_rcv_regid_ex_func_info_t_stct::store[LBM_UME_MAX_STORE_STRLEN]`

The store involved

**7.82.2.6** `lbm_ushort_t lbm_ume_rcv_regid_ex_func_info_t_stct::store_index`

The store index of the store involved

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.83 `lbm_ume_rcv_regid_ex_func_t_stct` Struct Reference

Structure that holds the application callback for registration ID setting, extended form.

```
#include <lbm.h>
```

### Data Fields

- [lbm\\_ume\\_rcv\\_regid\\_ex\\_function\\_cb](#) **func**
- `void * clientd`

#### 7.83.1 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:

- [lbm.h](#)

## 7.84 `lbm_ume_rcv_regid_func_t_stct` Struct Reference

Structure that holds the application callback for registration ID setting.

```
#include <lbm.h>
```

### Data Fields

- `lbm_ume_rcv_regid_function_cb` **func**
- `void * clientd`

#### 7.84.1 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:

- [lbm.h](#)

## 7.85 lbm\_ume\_src\_force\_reclaim\_func\_t\_stct Struct Reference

Structure that holds the application callback for forced reclamation notifications.

```
#include <lbm.h>
```

### Data Fields

- [lbm\\_ume\\_src\\_force\\_reclaim\\_function\\_cb](#) **func**
- void \* **clientd**

#### 7.85.1 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:

- [lbm.h](#)

## 7.86 `lbm_ume_store_entry_t_stct` Struct Reference

Structure that holds information for a UMP store for configuration purposes.

```
#include <lbm.h>
```

### Data Fields

- `lbm_uint_t ip_address`
- `lbm_ushort_t tcp_port`
- `lbm_ushort_t group_index`
- `lbm_uint_t registration_id`
- `lbm_uint32_t domain_id`

### 7.86.1 Detailed Description

A structure used with options to get/set information for a UMP store

### 7.86.2 Field Documentation

#### 7.86.2.1 `lbm_uint32_t lbm_ume_store_entry_t_stct::domain_id`

The domain ID of the store

#### 7.86.2.2 `lbm_ushort_t lbm_ume_store_entry_t_stct::group_index`

The index of the group this UMP store belongs to

#### 7.86.2.3 `lbm_uint_t lbm_ume_store_entry_t_stct::ip_address`

The IP address of the UMP store (network order)

#### 7.86.2.4 `lbm_uint_t lbm_ume_store_entry_t_stct::registration_id`

The registration ID that should be used with this UMP store for the source

#### 7.86.2.5 `lbm_ushort_t lbm_ume_store_entry_t_stct::tcp_port`

The TCP port of the store (network order)

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.87 `lbm_ume_store_group_entry_t_stct` Struct Reference

Structure that holds information for a UMP store group for configuration purposes.

```
#include <lbm.h>
```

### Data Fields

- `lbm_ushort_t` [index](#)
- `lbm_ushort_t` [group\\_size](#)

#### 7.87.1 Detailed Description

A structure used with options to get/set information for a UMP store group

#### 7.87.2 Field Documentation

##### 7.87.2.1 `lbm_ushort_t lbm_ume_store_group_entry_t_stct::group_size`

The size of the group

##### 7.87.2.2 `lbm_ushort_t lbm_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

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.88 `lbm_ume_store_name_entry_t_stct` Struct Reference

Structure that holds information for a UMP store by name for configuration purposes.

```
#include <lbm.h>
```

### Data Fields

- char `name` [LBM\_MAX\_CONTEXT\_NAME\_LEN+1]
- `lbm_ushort_t` `group_index`
- `lbm_uint_t` `registration_id`

#### 7.88.1 Detailed Description

A structure used with options to get/set information for a UMP store

#### 7.88.2 Field Documentation

##### 7.88.2.1 `lbm_ushort_t lbm_ume_store_name_entry_t_stct::group_index`

The index of the group this UMP store belongs to

##### 7.88.2.2 char `lbm_ume_store_name_entry_t_stct::name`[LBM\_MAX\_CONTEXT\_NAME\_LEN+1]

The store context name Restricted to alphanumeric characters, hyphens, and underscores

##### 7.88.2.3 `lbm_uint_t lbm_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:

- [lbm.h](#)

## 7.89 lbm\_umm\_info\_t\_stct Struct Reference

Structure for specifying UMM daemon connection options.

```
#include <lbm.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]
- lbm\_uint32\_t [flags](#)
- char \* [cert\\_file](#)
- char \* [cert\\_file\\_password](#)

### 7.89.1 Field Documentation

**7.89.1.1** char [lbm\\_umm\\_info\\_t\\_stct::appname](#)[LBM\_UMM\_APP\_NAME\_LENGTH\_MAX]

The UMM application name.

**7.89.1.2** char\* [lbm\\_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.

**7.89.1.3** char\* [lbm\\_umm\\_info\\_t\\_stct::cert\\_file\\_password](#)

Certificate file password. Required only if certificate file is password-protected.

**7.89.1.4** lbm\_uint32\_t [lbm\\_umm\\_info\\_t\\_stct::flags](#)

Flags, currently used to enable SSL.

**7.89.1.5** char [lbm\\_umm\\_info\\_t\\_stct::password](#)[LBM\_UMM\_PASSWORD\_LENGTH\_MAX]

The UMM user password.

**7.89.1.6** char [lbm\\_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.

**7.89.1.7** char [lbm\\_umm\\_info\\_t\\_stct::username](#)[LBM\_UMM\_USER\_NAME\_-LENGTH\_MAX]

The UMM user name.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.90 `lbm_umq_index_info_t_stct` Struct Reference

Structure that holds information used for sending and receiving messages with UMQ indices.

```
#include <lbm.h>
```

### Data Fields

- `int flags`
- `int reserved`
- `char index [LBM_UMQ_MAX_INDEX_LEN]`
- `size_t index_len`

### 7.90.1 Detailed Description

A structure used with UM sources and receivers to associated UMQ Indices with messages.

### 7.90.2 Field Documentation

#### 7.90.2.1 `int lbm_umq_index_info_t_stct::flags`

Flags that indicate optional portions are included

#### 7.90.2.2 `char lbm_umq_index_info_t_stct::index[LBM_UMQ_MAX_INDEX_LEN]`

The index

#### 7.90.2.3 `size_t lbm_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:

- `lbm.h`

## 7.91 `lbm_umq_msg_total_lifetime_info_t_stct` Struct Reference

Structure that holds UMQ message total lifetime information.

```
#include <lbm.h>
```

### Data Fields

- `int flags`
- `lbm_ulong_t umq_msg_total_lifetime`

#### 7.91.1 Detailed Description

A structure used with UMQ sources to specify a message's total lifetime.

#### 7.91.2 Field Documentation

##### 7.91.2.1 `int lbm_umq_msg_total_lifetime_info_t_stct::flags`

Flags that indicate which optional portions are included

##### 7.91.2.2 `lbm_ulong_t lbm_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:

- `lbm.h`

## 7.92 `lbm_umq_msgid_t_stct` Struct Reference

Structure that holds information for UMQ messages that allows the message to be identified uniquely.

```
#include <lbm.h>
```

### Data Fields

- [lbm\\_umq\\_regid\\_t](#) `regid`
- [lbm\\_uint64\\_t](#) `stamp`

#### 7.92.1 Detailed Description

See also:

[lbm\\_umq\\_regid\\_t](#) A structure used with UMQ messages to identify a message uniquely.

#### 7.92.2 Field Documentation

##### 7.92.2.1 [lbm\\_umq\\_regid\\_t lbm\\_umq\\_msgid\\_t\\_stct::regid](#)

The Registration ID of the original source context of the message

##### 7.92.2.2 [lbm\\_uint64\\_t lbm\\_umq\\_msgid\\_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:

- [lbm.h](#)

## 7.93 `lbm_umq_queue_entry_t_stct` Struct Reference

Structure that holds information for a UMQ queue registration ID for configuration purposes.

```
#include <lbm.h>
```

### Data Fields

- `char name` [LBM\_UMQ\_MAX\_QUEUE\_STRLEN]
- `lbm_umq_regid_t regid`

#### 7.93.1 Detailed Description

A struct used with options to get/set Registration ID information for UMQ queues

#### 7.93.2 Field Documentation

##### 7.93.2.1 `char lbm_umq_queue_entry_t_stct::name`[LBM\_UMQ\_MAX\_QUEUE\_STRLEN]

Name of the UMQ queue

##### 7.93.2.2 `lbm_umq_regid_t lbm_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:

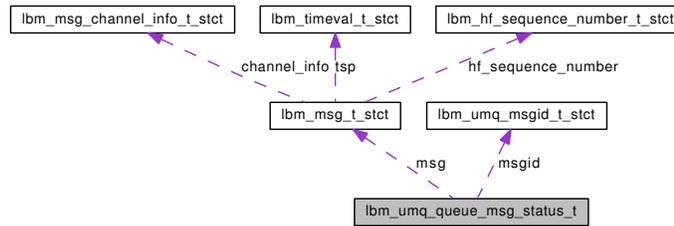
- `lbm.h`

## 7.94 lbm\_umq\_queue\_msg\_status\_t Struct Reference

Struct containing extended asynchronous operation status information about a single UMQ message.

```
#include <lbm.h>
```

Collaboration diagram for `lbm_umq_queue_msg_status_t`:



### Data Fields

- `lbm_umq_msgid_t msgid`
- `lbm_msg_t * msg`
- `void * clientd`
- `int status`
- `int flags`

### 7.94.1 Field Documentation

#### 7.94.1.1 `void* lbm_umq_queue_msg_status_t::clientd`

User client data pointer.

#### 7.94.1.2 `int lbm_umq_queue_msg_status_t::flags`

Reserved flags field.

#### 7.94.1.3 `lbm_msg_t* lbm_umq_queue_msg_status_t::msg`

Actual message, if appropriate and available (NULL otherwise)

#### 7.94.1.4 `lbm_umq_msgid_t lbm_umq_queue_msg_status_t::msgid`

UMQ message ID of this message.

**7.94.1.5** `int lbm_umq_queue_msg_status_t::status`

Status code for this message (retrieval in progress, consumed, etc.)

The documentation for this struct was generated from the following file:

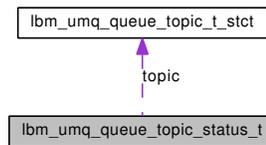
- [lbm.h](#)

## 7.95 `lbm_umq_queue_topic_status_t` Struct Reference

Struct containing extended asynchronous operation status information about a single UMQ topic.

```
#include <lbm.h>
```

Collaboration diagram for `lbm_umq_queue_topic_status_t`:



### Data Fields

- `lbm_umq_queue_topic_t * topic`
- `int status`
- `int flags`

#### 7.95.1 Field Documentation

##### 7.95.1.1 `int lbm_umq_queue_topic_status_t::flags`

Reserved flags field.

##### 7.95.1.2 `int lbm_umq_queue_topic_status_t::status`

Status code for this topic (reserved for future use; will currently always be set to 0).

##### 7.95.1.3 `lbm_umq_queue_topic_t* lbm_umq_queue_topic_status_t::topic`

UMQ topic info.

The documentation for this struct was generated from the following file:

- `lbm.h`

## 7.96 `lbm_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 <lbm.h>
```

### Data Fields

- char `topic_name` [LBM\_UMQ\_MAX\_TOPIC\_STRLEN]
- `lbm_umq_queue_application_set_t` \* `appsets`
- int `num_appsets`
- void \* `reserved`

### 7.96.1 Field Documentation

#### 7.96.1.1 `lbm_umq_queue_application_set_t`\* `lbm_umq_queue_topic_t_stct::appsets`

Array of application sets within this topic.

#### 7.96.1.2 int `lbm_umq_queue_topic_t_stct::num_appsets`

Length of the application sets array.

#### 7.96.1.3 void\* `lbm_umq_queue_topic_t_stct::reserved`

Reserved field; do not touch.

#### 7.96.1.4 char `lbm_umq_queue_topic_t_stct::topic_name`[LBM\_UMQ\_MAX\_TOPIC\_STRLEN]

The topic name string.

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.97 `lbm_umq_ulb_application_set_attr_t_stct` Struct Reference

Structure that holds information for a UMQ ULB sources application set attributes.

```
#include <lbm.h>
```

### Data Fields

- `lbm_ushort_t` [index](#)
- union {
  - int `d`
  - `lbm_ulong_t` `lu`
- } [value](#)

### 7.97.1 Detailed Description

A struct used with options to get/set UMQ ULB application set attributes

### 7.97.2 Field Documentation

#### 7.97.2.1 `int lbm_umq_ulb_application_set_attr_t_stct::d`

The integer value of the attribute

#### 7.97.2.2 `lbm_ushort_t lbm_umq_ulb_application_set_attr_t_stct::index`

Index of the Application Set

#### 7.97.2.3 `lbm_ulong_t lbm_umq_ulb_application_set_attr_t_stct::lu`

The unsigned long int value of the attribute

#### 7.97.2.4 `union { ... } lbm_umq_ulb_application_set_attr_t_stct::value`

The value of the attribute

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.98 `lbm_umq_ulb_receiver_type_attr_t_stct` Struct Reference

Structure that holds information for a UMQ ULB sources receiver type attributes.

```
#include <lbm.h>
```

### Data Fields

- `lbm_ulong_t id`
- union {
  - int `d`
  - `lbm_ulong_t lu`
- } `value`

### 7.98.1 Detailed Description

A struct used with options to get/set UMQ ULB receiver type attributes

### 7.98.2 Field Documentation

#### 7.98.2.1 `int lbm_umq_ulb_receiver_type_attr_t_stct::d`

The integer value of the attribute

#### 7.98.2.2 `lbm_ulong_t lbm_umq_ulb_receiver_type_attr_t_stct::id`

Receiver Type ID

#### 7.98.2.3 `lbm_ulong_t lbm_umq_ulb_receiver_type_attr_t_stct::lu`

The unsigned long int value of the attribute

#### 7.98.2.4 `union { ... } lbm_umq_ulb_receiver_type_attr_t_stct::value`

The value of the attribute

The documentation for this struct was generated from the following file:

- `lbm.h`

## 7.99 `lbm_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 <lbm.h>
```

### Data Fields

- `lbm_ulong_t id`
- `lbm_ushort_t application\_set\_index`

#### 7.99.1 Detailed Description

A struct used with options to get/set UMQ ULB Receiver Type entries

#### 7.99.2 Field Documentation

##### 7.99.2.1 `lbm_ushort_t lbm_umq_ulb_receiver_type_entry_t_stct::application_set_index`

Index of the Application Set the receiver is in

##### 7.99.2.2 `lbm_ulong_t lbm_umq_ulb_receiver_type_entry_t_stct::id`

Receiver Type ID

The documentation for this struct was generated from the following file:

- [lbm.h](#)

## 7.100 `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`

#### 7.100.1 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](#)

## 7.101 `lbm_wildcard_rcv_create_func_t_stct` Struct Reference

Structure that holds the receiver creation callback information for wildcard receivers.

```
#include <lbm.h>
```

### Data Fields

- [lbm\\_wildcard\\_rcv\\_create\\_function\\_cb](#) `createfunc`
- `void * clientd`

#### 7.101.1 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:

- [lbm.h](#)

## 7.102 `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

- [lbm\\_wildcard\\_rcv\\_delete\\_function\\_cb](#) `deletefunc`
- `void * clientd`

### 7.102.1 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:

- [lbm.h](#)

## 7.103 `lbm_wildcard_rcv_stats_t_stct` Struct Reference

Structure that holds statistics for a wildcard receiver.

```
#include <lbm.h>
```

### Data Fields

- char `pattern` [LBM\_MSG\_MAX\_TOPIC\_LEN]
- `lbm_uint8_t type`

#### 7.103.1 Detailed Description

THIS STRUCTURE IS UNSUPPORTED.

#### 7.103.2 Field Documentation

##### 7.103.2.1 char `lbm_wildcard_rcv_stats_t_stct::pattern`[LBM\_MSG\_MAX\_TOPIC\_LEN]

Pattern for the wildcard receiver. THIS FIELD IS UNSUPPORTED.

##### 7.103.2.2 `lbm_uint8_t lbm_wildcard_rcv_stats_t_stct::type`

Pattern type for the wildcard receiver. THIS FIELD IS UNSUPPORTED.

The documentation for this struct was generated from the following file:

- `lbm.h`

## 7.104 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](#)

#### 7.104.1 Field Documentation

##### 7.104.1.1 [lbm\\_ushort\\_t lbmmon\\_attr\\_block\\_t\\_stct::mEntryCount](#)

Number of attribute entries in network order.

##### 7.104.1.2 [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](#)

## 7.105 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 <lbmmon.h>
```

### Data Fields

- [lbm\\_ushort\\_t mType](#)
- [lbm\\_ushort\\_t mLength](#)

#### 7.105.1 Field Documentation

##### 7.105.1.1 lbm\_ushort\_t lbmmon\_attr\_entry\_t\_stct::mLength

Length of attribute data for this entry in network order.

##### 7.105.1.2 lbm\_ushort\_t lbmmon\_attr\_entry\_t\_stct::mType

Attribute type in network order.

The documentation for this struct was generated from the following file:

- [lbmmon.h](#)

## 7.106 lbmmon\_ctx\_statistics\_func\_t\_stct Struct Reference

A structure that holds the callback information for context statistics.

```
#include <lbmmon.h>
```

### Data Fields

- [lbmmon\\_ctx\\_statistics\\_cb cbfunc](#)

#### 7.106.1 Detailed Description

A structure used with receive controller options to get/set specific callback information.

#### 7.106.2 Field Documentation

##### 7.106.2.1 [lbmmon\\_ctx\\_statistics\\_cb lbmmon\\_ctx\\_statistics\\_func\\_t\\_stct::cbfunc](#)

Context statistics callback function.

The documentation for this struct was generated from the following file:

- [lbmmon.h](#)

## 7.107 `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

#### 7.107.1 Detailed Description

A structure used with receive controller options to get/set specific callback information.

#### 7.107.2 Field Documentation

##### 7.107.2.1 [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](#)

## 7.108 lbmmon\_format\_func\_t\_stct Struct Reference

Format module function pointer container.

```
#include <lbmmon.h>
```

### Data Fields

- [lbmmon\\_format\\_init\\_t](#) mInit
- [lbmmon\\_rcv\\_format\\_serialize\\_t](#) mRcvSerialize
- [lbmmon\\_src\\_format\\_serialize\\_t](#) mSrcSerialize
- [lbmmon\\_rcv\\_format\\_deserialize\\_t](#) mRcvDeserialize
- [lbmmon\\_src\\_format\\_deserialize\\_t](#) mSrcDeserialize
- [lbmmon\\_format\\_finish\\_t](#) mFinish
- [lbmmon\\_format\\_errmsg\\_t](#) mErrorMessage
- [lbmmon\\_evq\\_format\\_serialize\\_t](#) mEvqSerialize
- [lbmmon\\_evq\\_format\\_deserialize\\_t](#) mEvqDeserialize
- [lbmmon\\_ctx\\_format\\_serialize\\_t](#) mCtxSerialize
- [lbmmon\\_ctx\\_format\\_deserialize\\_t](#) mCtxDeserialize
- [lbmmon\\_rcv\\_topic\\_format\\_serialize\\_t](#) mRcvTopicSerialize
- [lbmmon\\_rcv\\_topic\\_format\\_deserialize\\_t](#) mRcvTopicDeserialize
- [lbmmon\\_wildcard\\_rcv\\_format\\_serialize\\_t](#) mWildcardRcvSerialize
- [lbmmon\\_wildcard\\_rcv\\_format\\_deserialize\\_t](#) mWildcardRcvDeserialize

### 7.108.1 Field Documentation

#### 7.108.1.1 [lbmmon\\_ctx\\_format\\_deserialize\\_t](#) lbmmon\_format\_func\_t\_stct::mCtxDeserialize

Function to deserialize context statistics (lbm\_context\_stats\_t).

#### 7.108.1.2 [lbmmon\\_ctx\\_format\\_serialize\\_t](#) lbmmon\_format\_func\_t\_stct::mCtxSerialize

Function to serialize context statistics (lbm\_context\_stats\_t).

#### 7.108.1.3 [lbmmon\\_format\\_errmsg\\_t](#) lbmmon\_format\_func\_t\_stct::mErrorMessage

Function to return a message describing the last error encountered.

**7.108.1.4 [lbmmon\\_evq\\_format\\_deserialize\\_t lbmmon\\_format\\_func\\_t\\_stct::mEvqDeserialize](#)**

Function to deserialize event queue statistics (`lbm_event_queue_stats_t`).

**7.108.1.5 [lbmmon\\_evq\\_format\\_serialize\\_t lbmmon\\_format\\_func\\_t\\_stct::mEvqSerialize](#)**

Function to serialize event queue statistics (`lbm_event_queue_stats_t`).

**7.108.1.6 [lbmmon\\_format\\_finish\\_t lbmmon\\_format\\_func\\_t\\_stct::mFinish](#)**

Format-specific cleanup function.

**7.108.1.7 [lbmmon\\_format\\_init\\_t lbmmon\\_format\\_func\\_t\\_stct::mInit](#)**

Initialization function.

**7.108.1.8 [lbmmon\\_rcv\\_format\\_deserialize\\_t lbmmon\\_format\\_func\\_t\\_stct::mRcvDeserialize](#)**

Function to deserialize receiver statistics (`lbm_rcv_transport_stats_t`).

**7.108.1.9 [lbmmon\\_rcv\\_format\\_serialize\\_t lbmmon\\_format\\_func\\_t\\_stct::mRcvSerialize](#)**

Function to serialize receiver statistics (`lbm_rcv_transport_stats_t`).

**7.108.1.10 [lbmmon\\_rcv\\_topic\\_format\\_deserialize\\_t lbmmon\\_format\\_func\\_t\\_stct::mRcvTopicDeserialize](#)**

Function to deserialize receiver topic statistics.

**7.108.1.11 [lbmmon\\_rcv\\_topic\\_format\\_serialize\\_t lbmmon\\_format\\_func\\_t\\_stct::mRcvTopicSerialize](#)**

Function to serialize receiver topic statistics.

**7.108.1.12** [lbmmon\\_src\\_format\\_deserialize\\_t lbmmon\\_format\\_func\\_t\\_stct::mSrcDeserialize](#)

Function to deserialize source statistics (lbm\_src\_transport\_stats\_t).

**7.108.1.13** [lbmmon\\_src\\_format\\_serialize\\_t lbmmon\\_format\\_func\\_t\\_stct::mSrcSerialize](#)

Function to serialize source statistics (lbm\_src\_transport\_stats\_t).

**7.108.1.14** [lbmmon\\_wildcard\\_rcv\\_format\\_deserialize\\_t lbmmon\\_format\\_func\\_t\\_stct::mWildcardRcvDeserialize](#)

Function to deserialize wildcard receiver statistics.

**7.108.1.15** [lbmmon\\_wildcard\\_rcv\\_format\\_serialize\\_t lbmmon\\_format\\_func\\_t\\_stct::mWildcardRcvSerialize](#)

Function to serialize wildcard receiver statistics.

The documentation for this struct was generated from the following file:

- [lbmmon.h](#)

## 7.109 lbmmon\_packet\_hdr\_t\_stct Struct Reference

Statistics packet header layout.

```
#include <lbmmon.h>
```

### Data Fields

- [lbm\\_uint\\_t mSignature](#)
- [lbm\\_ushort\\_t mType](#)
- [lbm\\_ushort\\_t mAttributeBlockLength](#)
- [lbm\\_ushort\\_t mDataLength](#)
- [lbm\\_ushort\\_t mFiller](#)

### 7.109.1 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.

### 7.109.2 Field Documentation

#### 7.109.2.1 lbm\_ushort\_t lbmmon\_packet\_hdr\_t\_stct::mAttributeBlockLength

Length of optional, variable-length attribute block in network order.

#### 7.109.2.2 lbm\_ushort\_t lbmmon\_packet\_hdr\_t\_stct::mDataLength

Length of variable-length statistics data in network order.

#### 7.109.2.3 lbm\_ushort\_t lbmmon\_packet\_hdr\_t\_stct::mFiller

Filler to assure proper alignment of the structure.

#### 7.109.2.4 lbm\_uint\_t lbmmon\_packet\_hdr\_t\_stct::mSignature

Packet signature in network order. Must be [LBMMON\\_PACKET\\_SIGNATURE](#).

### 7.109.2.5 lbm\_ushort\_t lbmmon\_packet\_hdr\_t\_stct::mType

Type of statistics, in network order. See LBMMON\_PACKET\_TYPE\_\*.

The documentation for this struct was generated from the following file:

- [lbmmon.h](#)

## 7.110 `lbmmon_rcv_statistics_func_t_stct` Struct Reference

A structure that holds the callback information for receiver statistics.

```
#include <lbmmon.h>
```

### Data Fields

- [lbmmon\\_rcv\\_statistics\\_cb](#) cbfunc

#### 7.110.1 Detailed Description

A structure used with receive controller options to get/set specific callback information.

#### 7.110.2 Field Documentation

##### 7.110.2.1 [lbmmon\\_rcv\\_statistics\\_cb](#) `lbmmon_rcv_statistics_func_t_stct::cbfunc`

Receiver statistics callback function.

The documentation for this struct was generated from the following file:

- [lbmmon.h](#)

## 7.111 lbmmon\_rcv\_topic\_statistics\_func\_t\_stct Struct Reference

For internal use only. A structure that holds the callback information for receiver topic statistics.

```
#include <lbmmon.h>
```

### Data Fields

- [lbmmon\\_rcv\\_topic\\_statistics\\_cb cbfunc](#)

#### 7.111.1 Detailed Description

A structure used with receive controller options to get/set specific callback information.

#### 7.111.2 Field Documentation

##### 7.111.2.1 [lbmmon\\_rcv\\_topic\\_statistics\\_cb lbmmon\\_rcv\\_topic\\_statistics\\_func\\_t\\_stct::cbfunc](#)

Receiver topic statistics callback function.

The documentation for this struct was generated from the following file:

- [lbmmon.h](#)

## 7.112 `lbmmon_src_statistics_func_t_stct` Struct Reference

A structure that holds the callback information for source statistics.

```
#include <lbmmon.h>
```

### Data Fields

- [lbmmon\\_src\\_statistics\\_cb](#) cbfunc

#### 7.112.1 Detailed Description

A structure used with receive controller options to get/set specific callback information.

#### 7.112.2 Field Documentation

##### 7.112.2.1 [lbmmon\\_src\\_statistics\\_cb](#) `lbmmon_src_statistics_func_t_stct::cbfunc`

Source statistics callback function.

The documentation for this struct was generated from the following file:

- [lbmmon.h](#)

## 7.113 lbmmon\_transport\_func\_t\_stct Struct Reference

Transport module function pointer container.

```
#include <lbmmon.h>
```

### Data Fields

- [lbmmon\\_transport\\_initsrc\\_t](#) mInitSource
- [lbmmon\\_transport\\_initrcv\\_t](#) mInitReceiver
- [lbmmon\\_transport\\_send\\_t](#) mSend
- [lbmmon\\_transport\\_receive\\_t](#) mReceive
- [lbmmon\\_transport\\_finishsrc\\_t](#) mFinishSource
- [lbmmon\\_transport\\_finishrcv\\_t](#) mFinishReceiver
- [lbmmon\\_transport\\_errmsg\\_t](#) mErrorMessage

### 7.113.1 Field Documentation

#### 7.113.1.1 [lbmmon\\_transport\\_errmsg\\_t](#) lbmmon\_transport\_func\_t\_stct::m-ErrorMessage

Function to return a message describing the last error encountered.

#### 7.113.1.2 [lbmmon\\_transport\\_finishrcv\\_t](#) lbmmon\_transport\_func\_t\_stct::m-FinishReceiver

Finish processing for a receiver transport.

#### 7.113.1.3 [lbmmon\\_transport\\_finishsrc\\_t](#) lbmmon\_transport\_func\_t\_stct::m-FinishSource

Finish processing for a source transport.

#### 7.113.1.4 [lbmmon\\_transport\\_initrcv\\_t](#) lbmmon\_transport\_func\_t\_stct::mInit-Receiver

Initialize module as a statistics receiver.

#### 7.113.1.5 [lbmmon\\_transport\\_initsrc\\_t](#) lbmmon\_transport\_func\_t\_stct::mInit-Source

Initialize module as a statistics source.

**7.113.1.6 [lbmmon\\_transport\\_receive\\_t](#) [lbmmon\\_transport\\_func\\_t\\_stct::mReceive](#)**

Receive statistics data.

**7.113.1.7 [lbmmon\\_transport\\_send\\_t](#) [lbmmon\\_transport\\_func\\_t\\_stct::mSend](#)**

Send a statistics packet.

The documentation for this struct was generated from the following file:

- [lbmmon.h](#)

## 7.114 lbmmon\_wildcard\_rcv\_statistics\_func\_t\_stct Struct Reference

A structure that holds the callback information for wildcard receiver statistics.

```
#include <lbmmon.h>
```

### Data Fields

- [lbmmon\\_wildcard\\_rcv\\_statistics\\_cb](#) cbfunc

#### 7.114.1 Detailed Description

A structure used with receive controller options to get/set specific callback information.

#### 7.114.2 Field Documentation

##### 7.114.2.1 [lbmmon\\_wildcard\\_rcv\\_statistics\\_cb](#) lbmmon\_wildcard\_rcv\_statistics\_func\_t\_stct::cbfunc

Wildcard receiver statistics callback function.

The documentation for this struct was generated from the following file:

- [lbmmon.h](#)

## 7.115 `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`

#### 7.115.1 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.

#### 7.115.2 Field Documentation

##### 7.115.2.1 `int8_t lbmpdm_decimal_t::exp`

Exponent.

##### 7.115.2.2 `int64_t lbmpdm_decimal_t::mant`

Mantissa.

The documentation for this struct was generated from the following file:

- `lbmpdm.h`

## 7.116 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]

### 7.116.1 Field Documentation

#### 7.116.1.1 [uint32\\_t lbmpdm\\_field\\_info\\_attr\\_stct\\_t::fixed\\_str\\_len](#)

The number of characters for fixed string or fixed unicode types.

#### 7.116.1.2 [uint32\\_t lbmpdm\\_field\\_info\\_attr\\_stct\\_t::num\\_arr\\_elem](#)

The number of elements for fixed arrays.

#### 7.116.1.3 [uint8\\_t lbmpdm\\_field\\_info\\_attr\\_stct\\_t::req](#)

If the field is required or not.

The documentation for this struct was generated from the following file:

- [lbmpdm.h](#)

## 7.117 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 \[256\]](#)

### 7.117.1 Field Documentation

#### 7.117.1.1 [uint16\\_t lbmpdm\\_field\\_value\\_stct\\_t::field\\_type](#)

The field type.

#### 7.117.1.2 [uint8\\_t lbmpdm\\_field\\_value\\_stct\\_t::is\\_array](#)

If the field is an array

#### 7.117.1.3 [uint8\\_t lbmpdm\\_field\\_value\\_stct\\_t::is\\_fixed](#)

If the field is a fixed length field.

#### 7.117.1.4 [size\\_t lbmpdm\\_field\\_value\\_stct\\_t::len](#)

The length in bytes of the field for scalar fields.

#### 7.117.1.5 [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.

**7.117.1.6** `uint32_t lbmpdm_field_value_stct_t::num_arr_elem`

The number of elements in the array for array fields.

**7.117.1.7** `char* lbmpdm_field_value_stct_t::value`

A pointer to the field value for scalar fields.

**7.117.1.8** `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](#)

## 7.118 `lbmpdm_timestamp_t` Struct Reference

Structure to hold a timestamp value.

```
#include <lbmpdm.h>
```

### Data Fields

- `int32_t tv_secs`
- `int32_t tv_usec`

#### 7.118.1 Detailed Description

The `tv_secs` is the number of seconds since the epoch. The `tv_usec` is the additional microseconds since the epoch, and can range from -128 to 127.

#### 7.118.2 Field Documentation

##### 7.118.2.1 `int32_t lbmpdm_timestamp_t::tv_secs`

Seconds since epoch

##### 7.118.2.2 `int32_t lbmpdm_timestamp_t::tv_usec`

Microseconds since last second

The documentation for this struct was generated from the following file:

- `lbmpdm.h`

## 7.119 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](#)

#### 7.119.1 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.

#### 7.119.2 Field Documentation

##### 7.119.2.1 [int8\\_t lbmsdm\\_decimal\\_t\\_stct::exp](#)

Exponent.

##### 7.119.2.2 [int64\\_t lbmsdm\\_decimal\\_t\\_stct::mant](#)

Mantissa.

The documentation for this struct was generated from the following file:

- [lbmsdm.h](#)

## 7.120 `ume_block_src_t_stct` Struct Reference

Structure used to designate an UME Block source.

```
#include <umeblocksrc.h>
```

### Data Fields

- `lbm_src_t * src`
- `lbm_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](#)

## 7.121 `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`

#### 7.121.1 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:

- [lbm.h](#)



# Chapter 8

## LBM API File Documentation

### 8.1 lbm.h File Reference

Ultra Messaging (UM) API.

```
#include <inttypes.h>
```

Include dependency graph for lbm.h:



#### Data Structures

- struct [lbm\\_iovec\\_t\\_stct](#)  
*Structure, struct iovec compatible, that holds information about buffers used for vectored sends.*
- struct [lbm\\_ipv4\\_address\\_mask\\_t\\_stct](#)  
*Structure that holds an IPv4 address and a CIDR style netmask.*
- struct [lbm\\_timeval\\_t\\_stct](#)  
*Structure that holds seconds and microseconds since midnight, Jan 1, 1970 UTC.*
- struct [lbm\\_src\\_event\\_wakeup\\_t\\_stct](#)  
*Structure that holds source wakeup event data.*
- struct [lbm\\_src\\_event\\_flight\\_size\\_notification\\_t\\_stct](#)

*Structure that holds flight size notification event data.*

- struct [lbm\\_src\\_event\\_ume\\_registration\\_t\\_stct](#)  
*Structure that holds store registration information for the UMP source.*
- struct [lbm\\_src\\_event\\_ume\\_registration\\_ex\\_t\\_stct](#)  
*Structure that holds store registration information for the UMP source in an extended form.*
- struct [lbm\\_src\\_event\\_ume\\_registration\\_complete\\_ex\\_t\\_stct](#)  
*Structure that holds information for sources after registration is complete to all involved stores.*
- struct [lbm\\_src\\_event\\_ume\\_deregistration\\_ex\\_t\\_stct](#)  
*Structure that holds store deregistration information for the UMP source in an extended form.*
- struct [lbm\\_msg\\_ume\\_registration\\_t\\_stct](#)  
*Structure that holds store registration information for the UMP receiver.*
- struct [lbm\\_msg\\_ume\\_registration\\_ex\\_t\\_stct](#)  
*Structure that holds store registration information for the UM receiver in an extended form.*
- struct [lbm\\_msg\\_ume\\_registration\\_complete\\_ex\\_t\\_stct](#)  
*Structure that holds information for receivers after registration is complete to all involved stores.*
- struct [lbm\\_msg\\_ume\\_deregistration\\_ex\\_t\\_stct](#)  
*Structure that holds store deregistration information for the UM receiver in an extended form.*
- struct [lbm\\_src\\_event\\_ume\\_ack\\_info\\_t\\_stct](#)  
*Structure that holds ACK information for a given message.*
- struct [lbm\\_src\\_event\\_ume\\_ack\\_ex\\_info\\_t\\_stct](#)  
*Structure that holds ACK information for a given message in an extended form.*
- struct [lbm\\_flight\\_size\\_inflight\\_t\\_stct](#)  
*Structure that holds information for source total inflight messages and bytes.*
- struct [lbm\\_umq\\_index\\_info\\_t\\_stct](#)  
*Structure that holds information used for sending and receiving messages with UMQ indices.*

- struct [lbm\\_msg\\_umq\\_index\\_assignment\\_eligibility\\_stop\\_complete\\_ex\\_t\\_stct](#)  
*Structure that holds index assignment information for receivers.*
- struct [lbm\\_msg\\_umq\\_index\\_assigned\\_ex\\_t\\_stct](#)  
*Structure that holds beginning-of-index information for receivers.*
- struct [lbm\\_msg\\_umq\\_index\\_released\\_ex\\_t\\_stct](#)  
*Structure that holds end-of-index information for receivers.*
- struct [lbm\\_msg\\_umq\\_index\\_assignment\\_eligibility\\_start\\_complete\\_ex\\_t\\_stct](#)  
*Structure that holds index assignment information for receivers.*
- struct [lbm\\_umq\\_msg\\_total\\_lifetime\\_info\\_t\\_stct](#)  
*Structure that holds UMQ message total lifetime information.*
- union [lbm\\_hf\\_sequence\\_number\\_t\\_stct](#)  
*Structure to hold a hot failover sequence number.*
- struct [lbm\\_umq\\_msgid\\_t\\_stct](#)  
*Structure that holds information for UMQ messages that allows the message to be identified uniquely.*
- struct [lbm\\_umq\\_queue\\_application\\_set\\_t\\_stct](#)
- struct [lbm\\_umq\\_queue\\_topic\\_t\\_stct](#)  
*Structure that holds queue topic information and can be used as a handle to a queue topic.*
- struct [lbm\\_umq\\_queue\\_topic\\_status\\_t](#)  
*Struct containing extended asynchronous operation status information about a single UMQ topic.*
- struct [lbm\\_ctx\\_umq\\_queue\\_topic\\_list\\_info\\_t](#)  
*Struct containing an array of queue topics retrieved via `lbm_umq_queue_topic_list`.*
- struct [lbm\\_umq\\_queue\\_msg\\_status\\_t](#)  
*Struct containing extended asynchronous operation status information about a single UMQ message.*
- struct [lbm\\_rcv\\_umq\\_queue\\_msg\\_list\\_info\\_t](#)  
*Struct containing an array of UMQ messages listed via `lbm_rcv_umq_queue_msg_list`.*

- struct [lbm\\_rcv\\_umq\\_queue\\_msg\\_retrieve\\_info\\_t](#)  
*Struct containing an array of UMQ messages retrieved via `lbm_rcv_umq_queue_msg_retrieve`.*
- struct [lbm\\_async\\_operation\\_info\\_t](#)  
*Results struct returned via the user-specified asynchronous operation callback from any asynchronous API.*
- struct [lbm\\_resolver\\_event\\_info\\_t\\_stct](#)  
*Resolver event structure (for internal use only).*
- struct [lbm\\_resolver\\_event\\_advertisement\\_t\\_stct](#)  
*Advertisement event structure (for internal use only).*
- struct [lbm\\_resolver\\_event\\_func\\_t\\_stct](#)  
*Resolver event function (for internal use only).*
- struct [lbm\\_async\\_operation\\_func\\_t](#)  
*Structure that holds information for asynchronous operation callbacks.*
- struct [lbm\\_src\\_send\\_ex\\_info\\_t\\_stct](#)  
*Structure that holds information for the extended send calls A structure used with UMP sources that utilize the extended send calls to pass options.*
- struct [lbm\\_ume\\_rcv\\_regid\\_ex\\_func\\_info\\_t\\_stct](#)  
*Structure that holds information for UMP receiver registration ID application callbacks.*
- struct [lbm\\_src\\_event\\_sequence\\_number\\_info\\_t\\_stct](#)  
*Structure that holds sequence number information for a message sent by a source.*
- struct [lbm\\_ume\\_rcv\\_recovery\\_info\\_ex\\_func\\_info\\_t\\_stct](#)  
*Structure that holds information for UMP receiver recovery sequence number info application callbacks.*
- struct [lbm\\_src\\_event\\_umq\\_message\\_id\\_info\\_t\\_stct](#)  
*Structure that holds Message ID information for a message sent by a sending UMQ application.*
- struct [lbm\\_context\\_event\\_umq\\_registration\\_ex\\_t\\_stct](#)  
*Structure that holds queue registration information for the UMQ context in an extended form.*
- struct [lbm\\_context\\_event\\_umq\\_registration\\_complete\\_ex\\_t\\_stct](#)

*Structure that holds information for contexts after registration is complete to all involved queue instances.*

- struct [lbm\\_src\\_event\\_umq\\_registration\\_complete\\_ex\\_t\\_stct](#)  
*Structure that holds information for sources after registration is complete to all involved queue instances.*
- struct [lbm\\_msg\\_umq\\_registration\\_complete\\_ex\\_t\\_stct](#)  
*Structure that holds information for receivers after registration is complete to all involved queue instances.*
- struct [lbm\\_src\\_event\\_umq\\_stability\\_ack\\_info\\_ex\\_t\\_stct](#)  
*Structure that holds UMQ ACK information for a given message in an extended form.*
- struct [lbm\\_msg\\_umq\\_deregistration\\_complete\\_ex\\_t\\_stct](#)  
*Structure that holds information for receivers after they de-register from a queue.*
- struct [lbm\\_src\\_event\\_umq\\_ulb\\_receiver\\_info\\_ex\\_t\\_stct](#)  
*Structure that holds UMQ ULB receiver information in an extended form.*
- struct [lbm\\_src\\_event\\_umq\\_ulb\\_message\\_info\\_ex\\_t\\_stct](#)  
*Structure that holds UMQ ULB message information in an extended form.*
- struct [lbm\\_str\\_hash\\_func\\_t\\_stct](#)
- struct [lbm\\_str\\_hash\\_func\\_ex\\_t\\_stct](#)  
*Structure that holds the hash function callback information.*
- struct [lbm\\_src\\_notify\\_func\\_t\\_stct](#)  
*Structure that holds the callback for source notifications.*
- struct [lbm\\_wildcard\\_rcv\\_compare\\_func\\_t\\_stct](#)  
*Structure that holds the application callback pattern type information for wildcard receivers.*
- struct [lbm\\_ume\\_rcv\\_regid\\_func\\_t\\_stct](#)  
*Structure that holds the application callback for registration ID setting.*
- struct [lbm\\_ume\\_rcv\\_regid\\_ex\\_func\\_t\\_stct](#)  
*Structure that holds the application callback for registration ID setting, extended form.*
- struct [lbm\\_ume\\_src\\_force\\_reclaim\\_func\\_t\\_stct](#)  
*Structure that holds the application callback for forced reclamation notifications.*

- struct [lbm\\_mim\\_unrecloss\\_func\\_t\\_stct](#)  
*Structure that holds the application callback for multicast immediate message unrecoverable loss notification.*
- struct [lbm\\_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\\_ucast\\_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 [lbm\\_serialized\\_response\\_t\\_stct](#)  
*Structure that holds a serialized UM response object.*
- struct [lbm\\_msg\\_fragment\\_info\\_t\\_stct](#)  
*Structure that holds fragment information for UM messages when appropriate.*
- struct [lbm\\_msg\\_gateway\\_info\\_t\\_stct](#)  
*Structure that holds originating information for UM messages which arrived via a gateway.*
- struct [lbm\\_msg\\_channel\\_info\\_t\\_stct](#)  
*Structure that represents UMS Spectrum channel information.*
- struct [lbm\\_msg\\_t\\_stct](#)  
*Structure that stores information about a received message.*
- struct [lbm\\_context\\_rcv\\_immediate\\_msgs\\_func\\_t\\_stct](#)  
*Structure that holds the application callback for receiving topic-less immediate mode messages.*
- struct [lbm\\_transport\\_source\\_info\\_t\\_stct](#)  
*Structure that holds formatted and parsed transport source strings.*
- struct [lbm\\_src\\_cost\\_func\\_t\\_stct](#)  
*Structure that holds the "source\_cost\_evaluation\_function" context attribute.*
- struct [lbm\\_config\\_option\\_stct\\_t](#)
- struct [lbm\\_wildcard\\_rcv\\_create\\_func\\_t\\_stct](#)  
*Structure that holds the receiver creation callback information for wildcard receivers.*
- struct [lbm\\_wildcard\\_rcv\\_delete\\_func\\_t\\_stct](#)  
*Structure that holds the receiver deletion callback information for wildcard receivers.*

- struct [lbm\\_src\\_transport\\_stats\\_tcp\\_t\\_stct](#)  
*Structure that holds datagram statistics for source TCP transports.*
- struct [lbm\\_src\\_transport\\_stats\\_lbtrm\\_t\\_stct](#)  
*Structure that holds datagram statistics for source LBT-RM transports.*
- struct [lbm\\_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\\_lbtsmx\\_t\\_stct](#)  
*Structure that holds datagram statistics for source LBT-SMX 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\\_lbtsmx\\_t\\_stct](#)  
*Structure that holds datagram statistics for receiver LBT-SMX transports.*

- struct [lbm\\_rcv\\_transport\\_stats\\_lbrdma\\_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\\_rcv\\_topic\\_stats\\_t\\_stct](#)  
*Structure that holds statistics for a receiver topic.*
- struct [lbm\\_wildcard\\_rcv\\_stats\\_t\\_stct](#)  
*Structure that holds statistics for a wildcard receiver.*
- struct [lbm\\_event\\_queue\\_cancel\\_cb\\_info\\_t\\_stct](#)  
*Structure passed to cancel/delete functions so that a cancel callback may be called.*
- struct [lbm\\_apphdr\\_chain\\_elem\\_t\\_stct](#)  
*Structure that represents an element in an app header chain.*
- struct [lbm\\_msg\\_properties\\_iter\\_t\\_stct](#)  
*A struct used for iterating over properties pointed to by an `lbm_msg_properties_t`.*
- struct [lbm\\_umm\\_info\\_t\\_stct](#)  
*Structure for specifying UMM daemon connection options.*

## Defines

- #define **LBM\_VERS\_MAJOR** 6
- #define **LBM\_VERS\_MINOR** 7
- #define **LBM\_VERS\_MAINT** 1
- #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 **LBMExpDLL**
- #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\_MSG\_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\_UME\_SRC\_REGID 0x4000
- #define LBM\_MSG\_FLAG\_NUMBERED\_CHANNEL 0x1
- #define LBM\_TOPIC\_RES\_REQUEST\_GW\_REMOTE\_INTEREST 0x40
- #define LBM\_TOPIC\_RES\_REQUEST\_CONTEXT\_QUERY 0x20
- #define LBM\_TOPIC\_RES\_REQUEST\_CONTEXT\_ADVERTISEMENT 0x10
- #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\_SRC\_EVENT\_UME\_MESSAGE\_NOT\_STABLE 33
- #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\_LBTSMX 0x4
- #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\_WINCPORT 4
- #define LBM\_CTX\_ATTR\_FDTYPE\_WINCPORT\_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\_WINRIOCPORT 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\_THRD\_POOL\_CREATE 1
- #define LBM\_CTX\_ATTR\_RCV\_THRD\_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 LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_3\_6\_5 0x00030605
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_0 0x00040000
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_0\_1 0x00040001
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_1 0x00040100
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_1\_1 0x00040101
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_1\_2 0x00040102
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_1\_3 0x00040103
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_LBM\_4\_2\_1 0x00040201

---

```

• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_2_-
  2 0x00040202
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_2_-
  3 0x00040203
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_2_-
  4 0x00040204
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_2_-
  5 0x00040205
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_2_-
  6 0x00040206
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_2_-
  7 0x00040207
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_2_-
  8 0x00040208
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_-
  0 0x01030000
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_0_-
  1 0x01030001
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_0_-
  2 0x01030002
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_-
  1 0x01030100
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_1_-
  1 0x01030101
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_1_-
  2 0x01030102
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_1_-
  3 0x01030103
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_2_-
  1 0x01030201
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_2_-
  2 0x01030202
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_2_-
  3 0x01030203
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_2_-
  4 0x01030204
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_2_-
  5 0x01030205
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_2_-
  6 0x01030206
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_2_-
  7 0x01030207
• #define      LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_2_-
  8 0x01030208

```

- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_1\_-  
0 0x02010000
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_1\_-  
1 0x02010100
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_1\_1\_-  
1 0x02010101
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_-  
0 0x02020000
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_0\_-  
1 0x02020001
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_1\_-  
1 0x02020101
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_1\_-  
3 0x02020103
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_1\_-  
4 0x02020104
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_1\_-  
5 0x02020105
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_1\_-  
6 0x02020106
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_1\_-  
7 0x02020107
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_1\_-  
8 0x02020108
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_1\_-  
9 0x02020109
- #define LBM\_CTX\_ATTR\_NET\_COMPAT\_MODE\_UMQ\_2\_1\_-  
10 0x0202010a
- #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\_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\_LBTSMX** LBM\_-  
TRANSPORT\_TYPE\_LBTSMX
- #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\_MAX\_UME\_STORES** 25
- #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\_MAX\_GATEWAY\_NAME\_LEN** 128
- #define **LBM\_OTID\_BLOCK\_SZ** 32
- #define **LBM\_CONTEXT\_INSTANCE\_BLOCK\_SZ** 8
- #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\_LBTSMX LBM\_TRANSPORT\_TYPE\_-  
LBTSMX
- #define LBM\_TRANSPORT\_STAT\_LBTRDMA LBM\_TRANSPORT\_-  
TYPE\_LBTRDMA
- #define LBM\_WILDCARD\_RCV\_PATTERN\_TYPE\_PCRC 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\_INTERGROUP\_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\_MESSAGE\_NOT\_STABLE\_FLAG\_-  
STORE 0x8
- #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_NOT\_STABLE\_FLAG\_-  
LOSS 0x40
- #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_NOT\_STABLE\_FLAG\_-  
TIMEOUT 0x80
- #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\_SUCCESS\_EX\_FLAG\_SRC\_-  
SID 0x8
- #define **LBM\_UME\_RCV\_RECOVERY\_INFO\_EX\_FLAG\_SRC\_SID** 0x1
- #define LBM\_MSG\_UME\_REGISTRATION\_COMPLETE\_EX\_FLAG\_-  
QUORUM 0x1
- #define LBM\_MSG\_UME\_REGISTRATION\_COMPLETE\_EX\_FLAG\_-  
RXREQMAX 0x2
- #define LBM\_MSG\_UME\_REGISTRATION\_COMPLETE\_EX\_FLAG\_-  
SRC\_SID 0x4
- #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\_UMQ\_MESSAGE\_STABLE\_EX\_FLAG\_INTRAGROUP\_STABLE 0x1
- #define LBM\_SRC\_EVENT\_UMQ\_MESSAGE\_STABLE\_EX\_FLAG\_INTERGROUP\_STABLE 0x2
- #define LBM\_SRC\_EVENT\_UMQ\_MESSAGE\_STABLE\_EX\_FLAG\_STABLE 0x4
- #define LBM\_SRC\_EVENT\_UMQ\_MESSAGE\_STABLE\_EX\_FLAG\_USER 0x8
- #define LBM\_MSG\_UMQ\_REGISTRATION\_COMPLETE\_EX\_FLAG\_QUORUM 0x1
- #define LBM\_MSG\_UMQ\_REGISTRATION\_COMPLETE\_EX\_FLAG\_ULB 0x2
- #define LBM\_MSG\_UMQ\_DEREGISTRATION\_COMPLETE\_EX\_FLAG\_ULB 0x1
- #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNED\_EX\_FLAG\_ULB 0x1
- #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNED\_EX\_FLAG\_REQUESTED 0x2
- #define LBM\_MSG\_UMQ\_INDEX\_RELEASED\_EX\_FLAG\_ULB 0x1
- #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNMENT\_ELIGIBILITY\_STOP\_COMPLETE\_EX\_FLAG\_ULB 0x1
- #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNMENT\_ELIGIBILITY\_START\_COMPLETE\_EX\_FLAG\_ULB 0x1
- #define LBM\_MSG\_UMQ\_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\_LBMMessageType** "LBMMessageType"
- #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\_RESOLVER\_EVENT\_ADVERTISEMENT\_TYPE** 0x01
- #define **LBM\_RESOLVER\_EVENT\_ADVERTISEMENT\_FLAGS\_LJ** 0x00000001
- #define **LBM\_RESOLVER\_EVENT\_ADVERTISEMENT\_FLAGS\_UME** 0x00000002

- #define **LBM\_RESOLVER\_EVENT\_ADVERTISEMENT\_FLAGS\_UMQ** 0x00000004
- #define **LBM\_RESOLVER\_EVENT\_ADVERTISEMENT\_FLAGS\_ULB** 0x00000008
- #define **LBM\_RESOLVER\_EVENT\_ADVERTISEMENT\_FLAGS\_EVC** 0x00000010
- #define **LBM\_RESOLVER\_EVENT\_INFO\_CAPABILITY\_VERSION\_FLAG\_UME** 0x1
- #define **LBM\_RESOLVER\_EVENT\_INFO\_CAPABILITY\_VERSION\_FLAG\_UMQ** 0x2
- #define **LBM\_RESOLVER\_EVENT\_INFO\_INFO\_DOMAIN\_ID\_PRESENT\_FLAG** 0x1ULL
- #define **LBM\_RESOLVER\_EVENT\_INFO\_INFO\_SOURCE\_ID\_PRESENT\_FLAG** 0x2ULL
- #define **LBM\_RESOLVER\_EVENT\_INFO\_INFO\_SOURCE\_ID\_TYPE\_PRESENT\_FLAG** 0x4ULL
- #define **LBM\_RESOLVER\_EVENT\_INFO\_INFO\_VERSION\_PRESENT\_FLAG** 0x8ULL
- #define **LBM\_RESOLVER\_EVENT\_INFO\_SRC\_TYPE\_APPLICATION** 0
- #define **LBM\_RESOLVER\_EVENT\_INFO\_SRC\_TYPE\_TNWGD** 1
- #define **LBM\_RESOLVER\_EVENT\_INFO\_SRC\_TYPE\_STORE** 2
- #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 **LBM\_RCV\_TOPIC\_STATS\_FLAG\_SRC\_VALID** 0x1
- #define **lbm\_rcv\_retrieve\_all\_transport\_stats**(r, n, s) lbm\_rcv\_retrieve\_all\_transport\_stats\_ex(r,n,sizeof(lbm\_rcv\_transport\_stats\_t),s)
- #define **lbm\_context\_retrieve\_rcv\_transport\_stats**(c, n, s) lbm\_context\_retrieve\_rcv\_transport\_stats\_ex(c,n,sizeof(lbm\_rcv\_transport\_stats\_t),s)
- #define **lbm\_context\_retrieve\_src\_transport\_stats**(c, n, s) lbm\_context\_retrieve\_src\_transport\_stats\_ex(c,n,sizeof(lbm\_src\_transport\_stats\_t),s)

## Typedefs

- typedef unsigned int **lbm\_uint\_t**
- typedef unsigned long int **lbm\_ulong\_t**

- typedef unsigned short int **lbm\_ushort\_t**
- typedef unsigned char **lbm\_uchar\_t**
- typedef uint8\_t **lbm\_uint8\_t**
- typedef uint16\_t **lbm\_uint16\_t**
- typedef uint32\_t **lbm\_uint32\_t**
- typedef uint64\_t **lbm\_uint64\_t**
- typedef int64\_t **lbm\_int64\_t**
- typedef int **lbm\_handle\_t**
- typedef lbm\_context\_t\_stct **lbm\_context\_t**
- typedef [lbm\\_iovec\\_t\\_stct](#) **lbm\_iovec\_t**  
*Structure, struct iovec compatible, that holds information about buffers used for vectored sends.*
- typedef [lbm\\_ipv4\\_address\\_mask\\_t\\_stct](#) **lbm\_ipv4\_address\_mask\_t**  
*Structure that holds an IPv4 address and a CIDR style netmask.*
- typedef [lbm\\_timeval\\_t\\_stct](#) **lbm\_timeval\_t**  
*Structure that holds seconds and microseconds since midnight, Jan 1, 1970 UTC.*
- typedef [lbm\\_src\\_event\\_wakeup\\_t\\_stct](#) **lbm\_src\_event\_wakeup\_t**  
*Structure that holds source wakeup event data.*
- typedef [lbm\\_src\\_event\\_flight\\_size\\_notification\\_t\\_stct](#) **lbm\_src\_event\_flight\_size\_notification\_t**  
*Structure that holds flight size notification event data.*
- typedef [lbm\\_src\\_event\\_ume\\_registration\\_t\\_stct](#) **lbm\_src\_event\_ume\_registration\_t**  
*Structure that holds store registration information for the UMP source.*
- typedef [lbm\\_src\\_event\\_ume\\_registration\\_ex\\_t\\_stct](#) **lbm\_src\_event\_ume\_registration\_ex\_t**  
*Structure that holds store registration information for the UMP source in an extended form.*
- typedef [lbm\\_src\\_event\\_ume\\_registration\\_complete\\_ex\\_t\\_stct](#) **lbm\_src\_event\_ume\_registration\_complete\_ex\_t**  
*Structure that holds information for sources after registration is complete to all involved stores.*
- typedef [lbm\\_src\\_event\\_ume\\_deregistration\\_ex\\_t\\_stct](#) **lbm\_src\_event\_ume\_deregistration\_ex\_t**  
*Structure that holds store deregistration information for the UMP source in an extended form.*

- typedef `lbm_msg_ume_registration_t_stct` `lbm_msg_ume_registration_t`  
*Structure that holds store registration information for the UMP receiver.*
- typedef `lbm_msg_ume_registration_ex_t_stct` `lbm_msg_ume_registration_ex_t`  
*Structure that holds store registration information for the UM receiver in an extended form.*
- typedef `lbm_msg_ume_registration_complete_ex_t_stct` `lbm_msg_ume_registration_complete_ex_t`  
*Structure that holds information for receivers after registration is complete to all involved stores.*
- typedef `lbm_msg_ume_deregistration_ex_t_stct` `lbm_msg_ume_deregistration_ex_t`  
*Structure that holds store deregistration information for the UM receiver in an extended form.*
- typedef `lbm_src_event_ume_ack_info_t_stct` `lbm_src_event_ume_ack_info_t`  
*Structure that holds ACK information for a given message.*
- typedef `lbm_src_event_ume_ack_ex_info_t_stct` `lbm_src_event_ume_ack_ex_info_t`  
*Structure that holds ACK information for a given message in an extended form.*
- typedef `lbm_flight_size_inflight_t_stct` `lbm_flight_size_inflight_t`  
*Structure that holds information for source total inflight messages and bytes.*
- typedef `lbm_src_channel_info_t_stct` `lbm_src_channel_info_t`
- typedef `lbm_umq_index_info_t_stct` `lbm_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` `lbm_msg_umq_index_assignment_eligibility_start_complete_ex_t`

*Structure that holds index assignment information for receivers.*

- typedef `lbm_apphdr_chain_t` `lbm_apphdr_chain_t`
- typedef `lbm_umq_msg_total_lifetime_info_t` `lbm_umq_msg_total_lifetime_info_t`

*Structure that holds UMQ message total lifetime information.*

- typedef `lbm_hf_sequence_number_t` `lbm_hf_sequence_number_t`

*Structure to hold a hot failover sequence number.*

- typedef `lbm_uint64_t` `lbm_umq_regid_t`
- typedef `lbm_umq_msgid_t` `lbm_umq_msgid_t`

*Structure that holds information for UMQ messages that allows the message to be identified uniquely.*

- typedef `lbm_umq_queue_application_set_t` `lbm_umq_queue_application_set_t`
- typedef `lbm_umq_queue_topic_t` `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` `lbm_msg_t`
- typedef `lbm_event_queue_t` `lbm_event_queue_t`
- typedef `lbm_uint64_t` `lbm_async_operation_handle_t`

*Opaque handle to an asynchronous operation.*

- typedef `lbm_resolver_event_info_t` `lbm_resolver_event_info_t`

*Resolver event structure (for internal use only).*

- typedef `lbm_uint32_t(*) lbm_resolver_event_cb_func` (`lbm_context_t *ctx`, `int event`, `const void *ed`, `const lbm_resolver_event_info_t *info`, `void *clientd`)

*Resolver event callback signature (for internal use only).*

- typedef `lbm_resolver_event_advertisement_t` `lbm_resolver_event_advertisement_t`

*Advertisement event structure (for internal use only).*

- typedef `lbm_resolver_event_func_t` `lbm_resolver_event_func_t`

*Resolver event function (for internal use only).*

- typedef int(\*) [lbm\\_async\\_operation\\_function\\_cb](#) ([lbm\\_async\\_operation\\_info\\_t](#) \*opinfo, void \*clientd)  
*User-supplied application callback for asynchronous operation status and completion.*
- typedef [lbm\\_msg\\_properties\\_t\\_stct](#) **lbm\_msg\_properties\_t**
- typedef [lbm\\_src\\_send\\_ex\\_info\\_t\\_stct](#) [lbm\\_src\\_send\\_ex\\_info\\_t](#)  
*Structure that holds information for the extended send calls A structure used with UM sources that utilize the extended send calls to pass options.*
- typedef [lbm\\_ume\\_rcv\\_regid\\_ex\\_func\\_info\\_t\\_stct](#) [lbm\\_ume\\_rcv\\_regid\\_ex\\_func\\_info\\_t](#)  
*Structure that holds information for UMP receiver registration ID application callbacks.*
- typedef [lbm\\_src\\_event\\_sequence\\_number\\_info\\_t\\_stct](#) [lbm\\_src\\_event\\_sequence\\_number\\_info\\_t](#)  
*Structure that holds sequence number information for a message sent by a source.*
- typedef [lbm\\_ume\\_rcv\\_recovery\\_info\\_ex\\_func\\_info\\_t\\_stct](#) [lbm\\_ume\\_rcv\\_recovery\\_info\\_ex\\_func\\_info\\_t](#)  
*Structure that holds information for UMP receiver recovery sequence number info application callbacks.*
- typedef [lbm\\_src\\_event\\_umq\\_message\\_id\\_info\\_t\\_stct](#) [lbm\\_src\\_event\\_umq\\_message\\_id\\_info\\_t](#)  
*Structure that holds Message ID information for a message sent by a sending UMQ application.*
- typedef [lbm\\_context\\_event\\_umq\\_registration\\_ex\\_t\\_stct](#) [lbm\\_context\\_event\\_umq\\_registration\\_ex\\_t](#)  
*Structure that holds queue registration information for the UMQ context in an extended form.*
- typedef [lbm\\_context\\_event\\_umq\\_registration\\_complete\\_ex\\_t\\_stct](#) [lbm\\_context\\_event\\_umq\\_registration\\_complete\\_ex\\_t](#)  
*Structure that holds information for contexts after registration is complete to all involved queue instances.*
- typedef [lbm\\_src\\_event\\_umq\\_registration\\_complete\\_ex\\_t\\_stct](#) [lbm\\_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(\*) [lbm\\_ume\\_src\\_force\\_reclaim\\_function\\_cb](#) (const char \*topic\_str, lbm\_uint\_t seqnum, void \*clientd)

*Application callback for notification of forced reclamation of retained messages for UMP sources.*

- typedef int(\*) [lbm\\_mim\\_unrecloss\\_function\\_cb](#) (const char \*source\_name, lbm\_uint\_t seqnum, void \*clientd)

*Application callback in receiving application for notification of unrecoverable lost messages from a multicast immediate message sender.*

- typedef int(\*) [lbm\\_ume\\_rcv\\_recovery\\_info\\_ex\\_function\\_cb](#) ([lbm\\_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 (\*)(\*) [lbm\\_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(\*) [lbm\\_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 lbm\_str\_hash\_func\_t\_stct [lbm\\_str\\_hash\\_func\\_t](#)
- typedef [lbm\\_str\\_hash\\_func\\_ex\\_t\\_stct](#) [lbm\\_str\\_hash\\_func\\_ex\\_t](#)

*Structure that holds the hash function callback information.*

- typedef [lbm\\_src\\_notify\\_func\\_t\\_stct](#) [lbm\\_src\\_notify\\_func\\_t](#)

*Structure that holds the callback for source notifications.*

- typedef [lbm\\_wildcard\\_rcv\\_compare\\_func\\_t\\_stct](#) [lbm\\_wildcard\\_rcv\\_compare\\_func\\_t](#)

*Structure that holds the application callback pattern type information for wildcard receivers.*

- typedef [lbm\\_ume\\_rcv\\_regid\\_func\\_t\\_stct](#) [lbm\\_ume\\_rcv\\_regid\\_func\\_t](#)

*Structure that holds the application callback for registration ID setting.*

- typedef [lbm\\_ume\\_rcv\\_regid\\_ex\\_func\\_t\\_stct](#) [lbm\\_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\\_rcv\\_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 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\_rcv\_t\_stct **lbm\_hfx\_rcv\_t**
- typedef lbm\_topic\_t\_stct **lbm\_topic\_t**
- typedef lbm\_src\_t\_stct **lbm\_src\_t**
- typedef lbm\_rcv\_t\_stct **lbm\_rcv\_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\_rcv\_topic\_attr\_t\_stct **lbm\_rcv\_topic\_attr\_t**

- typedef int(\*) [lbm\\_wildcard\\_rcv\\_create\\_function\\_cb](#) (const char \*topic\_str, lbm\_rcv\_topic\_attr\_t \*attr, void \*clientd)  
*Application callback for wildcard receiver creation.*
- typedef [lbm\\_wildcard\\_rcv\\_create\\_func\\_t\\_stct](#) [lbm\\_wildcard\\_rcv\\_create\\_func\\_t](#)  
*Structure that holds the receiver creation callback information for wildcard receivers.*
- typedef int(\*) [lbm\\_wildcard\\_rcv\\_delete\\_function\\_cb](#) (const char \*topic\_str, void \*clientd)  
*Application callback for wildcard receiver deletion.*
- typedef [lbm\\_wildcard\\_rcv\\_delete\\_func\\_t\\_stct](#) [lbm\\_wildcard\\_rcv\\_delete\\_func\\_t](#)  
*Structure that holds the receiver deletion callback information for wildcard receivers.*
- typedef lbm\_wildcard\_rcv\_attr\_t\_stct **lbm\_wildcard\_rcv\_attr\_t**
- typedef [lbm\\_src\\_transport\\_stats\\_tcp\\_t\\_stct](#) [lbm\\_src\\_transport\\_stats\\_tcp\\_t](#)  
*Structure that holds datagram statistics for source TCP transports.*
- typedef [lbm\\_src\\_transport\\_stats\\_lbtrm\\_t\\_stct](#) [lbm\\_src\\_transport\\_stats\\_lbtrm\\_t](#)  
*Structure that holds datagram statistics for source LBT-RM transports.*
- typedef [lbm\\_src\\_transport\\_stats\\_daemon\\_t\\_stct](#) [lbm\\_src\\_transport\\_stats\\_daemon\\_t](#)  
*Structure that holds statistics for source daemon mode transport (deprecated).*
- typedef [lbm\\_src\\_transport\\_stats\\_lbtru\\_t\\_stct](#) [lbm\\_src\\_transport\\_stats\\_lbtru\\_t](#)  
*Structure that holds datagram statistics for source LBT-RU transports.*
- typedef [lbm\\_src\\_transport\\_stats\\_lbtipc\\_t\\_stct](#) [lbm\\_src\\_transport\\_stats\\_lbtipc\\_t](#)  
*Structure that holds datagram statistics for source LBT-IPC transports.*
- typedef [lbm\\_src\\_transport\\_stats\\_lbtsmx\\_t\\_stct](#) [lbm\\_src\\_transport\\_stats\\_lbtsmx\\_t](#)  
*Structure that holds datagram statistics for source LBT-SMX transports.*
- typedef [lbm\\_src\\_transport\\_stats\\_lbrdma\\_t\\_stct](#) [lbm\\_src\\_transport\\_stats\\_lbrdma\\_t](#)  
*Structure that holds datagram statistics for source LBT-RDMA transports.*
- typedef [lbm\\_src\\_transport\\_stats\\_t\\_stct](#) [lbm\\_src\\_transport\\_stats\\_t](#)  
*Structure that holds statistics for source transports.*

- typedef [lbm\\_rcv\\_transport\\_stats\\_tcp\\_t\\_stct](#) [lbm\\_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\\_lbtsmx\\_t\\_stct](#) [lbm\\_rcv\\_transport\\_stats\\_lbtsmx\\_t](#)  
*Structure that holds datagram statistics for receiver LBT-SMX 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 [lbm\\_rcv\\_topic\\_stats\\_t\\_stct](#) [lbm\\_rcv\\_topic\\_stats\\_t](#)  
*Structure that holds statistics for a receiver topic.*
- typedef [lbm\\_wildcard\\_rcv\\_stats\\_t\\_stct](#) [lbm\\_wildcard\\_rcv\\_stats\\_t](#)  
*Structure that holds statistics for a wildcard receiver.*
- 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(\*) [lbm\\_request\\_cb\\_proc](#) (lbm\_request\_t \*req, [lbm\\_msg\\_t](#) \*msg, void \*clientd)  
*Application callback for responses returned when a request is sent.*
- 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)  
*Application callback for event queue monitor events.*
- typedef int(\*) [lbm\\_log\\_cb\\_proc](#) (int level, const char \*message, void \*clientd)  
*Application callback for message logging.*
- typedef int(\*) [lbm\\_daemon\\_event\\_cb\\_proc](#) (lbm\_context\_t \*ctx, int event, const char \*info, void \*clientd)  
*Application callback for daemon events.*
- typedef void(\*) [lbm\\_event\\_queue\\_cancel\\_cb\\_proc](#) (int dispatch\_thrd, void \*clientd)  
*Application callback for lbm\_\*\_delete\_ex().*
- typedef [lbm\\_event\\_queue\\_cancel\\_cb\\_info\\_t\\_stct](#) [lbm\\_event\\_queue\\_cancel\\_cb\\_info\\_t](#)  
*Structure passed to cancel/delete functions so that a cancel callback may be called.*
- typedef int(\*) [lbm\\_flight\\_size\\_set\\_inflight\\_cb\\_proc](#) (int inflight, void \*clientd)  
*Application callback for lbm\_\*\_flight\_size\_set\_inflight().*
- typedef void(\*) [lbm\\_flight\\_size\\_set\\_inflight\\_ex\\_cb\\_proc](#) ([lbm\\_flight\\_size\\_inflight\\_t](#) \*inflight, void \*clientd)  
*Application callback for lbm\_ume\_flight\_size\_set\_inflight\_ex(). Change the inflight parameter messages and bytes to update the current settings.*

- typedef lbm\_apphdr\_chain\_iter\_t\_stct **lbm\_apphdr\_chain\_iter\_t**
- typedef lbm\_apphdr\_chain\_elem\_t\_stct **lbm\_apphdr\_chain\_elem\_t**  
*Structure that represents an element in an app header chain.*
- typedef lbm\_msg\_properties\_iter\_t\_stct **lbm\_msg\_properties\_iter\_t**  
*A struct used for iterating over properties pointed to by an lbm\_msg\_properties\_t.*
- typedef lbm\_umq\_msg\_selector\_t\_stct **lbm\_umq\_msg\_selector\_t**
- typedef int(\*) **lbm\_cred\_callback\_fn** (const char \*name, size\_t name\_len, const char \*passwd, size\_t passwd\_len, void \*clientd)
- typedef lbm\_umm\_info\_t\_stct **lbm\_umm\_info\_t**  
*Structure for specifying UMM daemon connection options.*

## Enumerations

- enum { **LBM\_OK** = 0, **LBM\_FAILURE** = -1 }

## Functions

- LBMEExpDLL const char \* **lbm\_version** (void)  
*return the version string compiled into UM.*
- LBMEExpDLL int **lbm\_context\_dump** (lbm\_context\_t \*ctx, int \*size, lbm\_config\_option\_t \*opts)  
*Retrieves all context attribute options.*
- LBMEExpDLL int **lbm\_context\_attr\_dump** (lbm\_context\_attr\_t \*cattr, int \*size, lbm\_config\_option\_t \*opts)  
*Retrieves all context attribute options.*
- LBMEExpDLL int **lbm\_context\_attr\_option\_size** ()  
*Retrieves the number of options that are of type "context".*
- LBMEExpDLL int **lbm\_context\_attr\_create** (lbm\_context\_attr\_t \*\*attr)  
*Create and fill a UM context attribute object with the current default values.*
- LBMEExpDLL int **lbm\_context\_attr\_create\_default** (lbm\_context\_attr\_t \*\*attr)  
*Create and fill a UM context attribute object with the initial default values.*

- LBMEpDLL int [lbm\\_context\\_attr\\_create\\_from\\_xml](#) (lbm\_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.*
- LBMEpDLL int [lbm\\_context\\_attr\\_set\\_from\\_xml](#) (lbm\_context\_attr\_t \*attr, const char \*context\_name)  
*Fill a UM context attribute object with the current default values for the given context name.*
- LBMEpDLL int [lbm\\_context\\_attr\\_delete](#) (lbm\_context\_attr\_t \*attr)  
*Delete a UM context attribute object.*
- LBMEpDLL int [lbm\\_context\\_attr\\_dup](#) (lbm\_context\_attr\_t \*\*attr, const lbm\_context\_attr\_t \*original)  
*Duplicate a UM context attribute object.*
- LBMEpDLL int [lbm\\_context\\_attr\\_setopt](#) (lbm\_context\_attr\_t \*attr, const char \*optname, const void \*optval, size\_t optlen)  
*Set an option for the given UM context attribute.*
- LBMEpDLL int [lbm\\_context\\_attr\\_str\\_setopt](#) (lbm\_context\_attr\_t \*attr, const char \*optname, const char \*optval)  
*Set an option for the given UM context attribute using a string.*
- LBMEpDLL int [lbm\\_context\\_attr\\_getopt](#) (lbm\_context\_attr\_t \*attr, const char \*optname, void \*optval, size\_t \*optlen)  
*Retrieve the value of an option for the given UM context attribute.*
- LBMEpDLL int [lbm\\_context\\_attr\\_str\\_getopt](#) (lbm\_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.*
- LBMEpDLL int [lbm\\_context\\_create](#) (lbm\_context\_t \*\*ctxp, const lbm\_context\_attr\_t \*attr, [lbm\\_daemon\\_event\\_cb\\_proc](#) proc, void \*clientd)  
*Create and initialize an lbm\_context\_t object.*
- LBMEpDLL int [lbm\\_context\\_reactor\\_only\\_create](#) (lbm\_context\_t \*\*ctxp, const lbm\_context\_attr\_t \*attr)  
*Create and initialize an lbm\_context\_t object suitable for FD and timers only.*
- LBMEpDLL int [lbm\\_context\\_delete](#) (lbm\_context\_t \*ctx)  
*Delete a UM context object.*

- LBMEExpDLL int [lbm\\_context\\_delete\\_ex](#) (lbm\_context\_t \*ctx, [lbm\\_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.*

- LBMEExpDLL int [lbm\\_context\\_topic\\_resolution\\_request](#) (lbm\_context\_t \*ctx, lbm\_ushort\_t flags, lbm\_ulong\_t interval\_msec, lbm\_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 [lbm\\_context\\_setopt](#) (lbm\_context\_t \*ctx, const char \*optname, const void \*optval, size\_t optlen)

*Set an option value within the given ctx.*

- LBMEExpDLL int [lbm\\_context\\_str\\_setopt](#) (lbm\_context\_t \*ctx, const char \*optname, const char \*optval)

*Set an option value within the given ctx.*

- LBMEExpDLL int [lbm\\_context\\_getopt](#) (lbm\_context\_t \*ctx, const char \*optname, void \*optval, size\_t \*optlen)

*Retrieve an option value within the given ctx.*

- LBMEExpDLL int [lbm\\_context\\_str\\_getopt](#) (lbm\_context\_t \*ctx, const char \*optname, char \*optval, size\_t \*optlen)

*Retrieve the textual option value within the given ctx.*

- LBMEExpDLL int [lbm\\_context\\_rcv\\_immediate\\_msgs](#) (lbm\_context\_t \*ctx, [lbm\\_immediate\\_msg\\_cb\\_proc](#) proc, void \*clientd, lbm\_event\_queue\_t \*evq)

*Set the callback procedure and delivery method for non-topic immediate messages.*

- 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)

*Set the callback procedure and delivery method for immediate messages to a topic for which there is no receiver.*

- LBMEpDLL int [lbm\\_context\\_set\\_name](#) (lbm\_context\_t \*ctx, const char \*name)  
*Set the name associated with a context.*
- LBMEpDLL int [lbm\\_context\\_get\\_name](#) (lbm\_context\_t \*ctx, char \*name, size\_t \*size)  
*Get the name associated with a context.*
- LBMEpDLL int [lbm\\_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.*
- LBMEpDLL int [lbm\\_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.*
- LBMEpDLL int [lbm\\_license\\_ummmn\\_valid](#) ()  
*Determine is the MnM product is licensed.*
- LBMEpDLL int [lbm\\_license\\_vds\\_valid](#) ()  
*Determine is the VDS product is licensed.*
- LBMEpDLL int [lbm\\_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.*
- LBMEpDLL int [lbm\\_config\\_xml\\_file](#) (const char \*url, const char \*application\_name)  
*Load a UM XML configuration file.*
- LBMEpDLL int [lbm\\_config\\_xml\\_string](#) (const char \*xml\_data, const char \*application\_name)  
*Load UM XML configuration data.*
- LBMEpDLL int [lbm\\_log](#) (lbm\_log\_cb\_proc proc, void \*clientd)  
*Set a callback function to be called for UM log messages (warnings, notices, etc.).*
- LBMEpDLL void [lbm\\_logf](#) (int level, const char \*format,...)  
*Log a message. This is an entry to the UM logging mechanism.*
- LBMEpDLL const char \* [lbm\\_errmsg](#) (void)  
*Return an ASCII string containing the error message last encountered by this thread.*

- LBMEpDLL int [lbm\\_errnum](#) (void)  
*Return the error number last encountered by this thread.*
- LBMEpDLL int [lbm\\_win32\\_static\\_thread\\_attach](#) (void)  
*Instructs UM that a new thread will be calling UM functions.*
- LBMEpDLL int [lbm\\_win32\\_static\\_thread\\_detach](#) (void)  
*Instructs UM that a new thread is done calling UM functions.*
- LBMEpDLL int [lbm\\_schedule\\_timer](#) (lbm\_context\_t \*ctx, [lbm\\_timer\\_cb\\_proc](#) proc, void \*clientd, lbm\_event\_queue\_t \*evq, lbm\_ulong\_t delay)  
*Schedule a timer that calls proc when it expires.*
- LBMEpDLL 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)  
*Schedule a recurring timer that calls proc when it expires.*
- LBMEpDLL int [lbm\\_cancel\\_timer](#) (lbm\_context\_t \*ctx, int id, void \*\*clientdp)  
*Cancel a previously scheduled timer identified by id.*
- LBMEpDLL int [lbm\\_cancel\\_timer\\_ex](#) (lbm\_context\_t \*ctx, int id, void \*\*clientdp, [lbm\\_event\\_queue\\_cancel\\_cb\\_info\\_t](#) \*cbinfo)  
*Extended cancel a previously scheduled timer identified by id.*
- LBMEpDLL int [lbm\\_context\\_process\\_events](#) (lbm\_context\_t \*ctx, lbm\_ulong\_t msec)  
*Process internal events in the given UM context object.*
- LBMEpDLL int [lbm\\_context\\_unblock](#) (lbm\_context\_t \*ctx)  
*Unblock a sequential mode UM context.*
- LBMEpDLL int [lbm\\_context\\_process\\_lbtipc\\_messages](#) (lbm\_context\_t \*ctx, lbm\_ulong\_t msec, lbm\_ulong\_t loop\_count)  
*Process LBT-IPC messages received.*
- LBMEpDLL int [lbm\\_context\\_lbtipc\\_unblock](#) (lbm\_context\_t \*ctx)  
*Unblock a sequential mode LBT-IPC processing loop.*
- LBMEpDLL 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)

*Register a file descriptor/socket for events that calls proc when a given event occurs.*

- LBMEpDLL int `lbm_cancel_fd` (lbm\_context\_t \*ctx, lbm\_handle\_t handle, lbm\_ulong\_t ev)  
*Cancel a previously registered file descriptor/socket event.*
- LBMEpDLL 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)  
*Extended cancel a previously registered file descriptor/socket event.*
- LBMEpDLL int `lbm_src_topic_dump` (lbm\_src\_t \*src, int \*size, lbm\_config\_option\_t \*opts)  
*Retrieves all source topic attribute options.*
- LBMEpDLL int `lbm_src_topic_attr_dump` (lbm\_src\_topic\_attr\_t \*sattr, int \*size, lbm\_config\_option\_t \*opts)  
*Retrieves all source topic attribute options.*
- LBMEpDLL int `lbm_src_topic_attr_option_size` ()  
*Retrieves the number of options that are of type "topic".*
- LBMEpDLL int `lbm_src_topic_alloc` (lbm\_topic\_t \*\*topicp, lbm\_context\_t \*ctx, const char \*symbol, const lbm\_src\_topic\_attr\_t \*attr)  
*Turn a Topic string into a UM topic object usable by sources.*
- LBMEpDLL int `lbm_src_topic_attr_create` (lbm\_src\_topic\_attr\_t \*\*attr)  
*Create and fill a UM source topic attribute object with the current default values.*
- LBMEpDLL int `lbm_src_topic_attr_create_default` (lbm\_src\_topic\_attr\_t \*\*attr)  
*Create and fill a UM source topic attribute object with the initial default values.*
- LBMEpDLL int `lbm_src_topic_attr_create_from_xml` (lbm\_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.*
- LBMEpDLL int `lbm_src_topic_attr_set_from_xml` (lbm\_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.*
- LBMEpDLL int `lbm_src_topic_attr_delete` (lbm\_src\_topic\_attr\_t \*attr)

*Delete a source topic attribute object.*

- LBMEExpDLL int [lbm\\_src\\_topic\\_attr\\_dup](#) (lbm\_src\_topic\_attr\_t \*\*attr, const lbm\_src\_topic\_attr\_t \*original)

*Duplicate a UM source topic attribute object.*

- LBMEExpDLL int [lbm\\_src\\_topic\\_attr\\_setopt](#) (lbm\_src\_topic\_attr\_t \*attr, const char \*optname, const void \*optval, size\_t optlen)

*Set an option value within the given source topic attribute.*

- LBMEExpDLL int [lbm\\_src\\_topic\\_attr\\_str\\_setopt](#) (lbm\_src\_topic\_attr\_t \*attr, const char \*optname, const char \*optval)

*Set an option value within the given source topic attribute.*

- LBMEExpDLL int [lbm\\_src\\_topic\\_attr\\_getopt](#) (lbm\_src\_topic\_attr\_t \*attr, const char \*optname, void \*optval, size\_t \*optlen)

*Retrieve an option value within the given source topic attribute.*

- LBMEExpDLL int [lbm\\_src\\_topic\\_attr\\_str\\_getopt](#) (lbm\_src\_topic\_attr\_t \*attr, const char \*optname, char \*optval, size\_t \*optlen)

*Retrieve a textual option value within the given source topic attribute.*

- LBMEExpDLL int [lbm\\_rcv\\_topic\\_lookup](#) (lbm\_topic\_t \*\*topicp, lbm\_context\_t \*ctx, const char \*symbol, const lbm\_rcv\_topic\_attr\_t \*attr)

*Turn a Topic string into a UM topic object usable by receivers.*

- LBMEExpDLL int [lbm\\_rcv\\_topic\\_dump](#) (lbm\_rcv\_t \*rcv, int \*size, lbm\_config\_option\_t \*opts)

*Retrieves all receiver topic attribute options.*

- LBMEExpDLL int [lbm\\_rcv\\_topic\\_attr\\_dump](#) (lbm\_rcv\_topic\_attr\_t \*rattr, int \*size, lbm\_config\_option\_t \*opts)

*Retrieves all receiver topic attribute options.*

- LBMEExpDLL int [lbm\\_rcv\\_topic\\_attr\\_option\\_size](#) ()

*Retrieves the number of options that are of type "source topic".*

- LBMEExpDLL int [lbm\\_rcv\\_topic\\_attr\\_create](#) (lbm\_rcv\_topic\_attr\_t \*\*attr)

*Create and fill a UM receiver topic attribute object with the current default values.*

- LBMEExpDLL int [lbm\\_rcv\\_topic\\_attr\\_create\\_default](#) (lbm\_rcv\_topic\_attr\_t \*\*attr)

*Create and fill a UM receiver topic attribute object with the initial default values.*

- LBMEExpDLL int [lbm\\_rcv\\_topic\\_attr\\_create\\_from\\_xml](#) (lbm\_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 [lbm\\_rcv\\_topic\\_attr\\_set\\_from\\_xml](#) (lbm\_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 [lbm\\_rcv\\_topic\\_attr\\_delete](#) (lbm\_rcv\_topic\_attr\_t \*attr)  
*Delete a receiver topic attribute object.*
- LBMEExpDLL int [lbm\\_rcv\\_topic\\_attr\\_dup](#) (lbm\_rcv\_topic\_attr\_t \*\*attr, const lbm\_rcv\_topic\_attr\_t \*original)  
*Duplicate a UM receiver topic attribute object.*
- LBMEExpDLL int [lbm\\_rcv\\_topic\\_attr\\_setopt](#) (lbm\_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 [lbm\\_rcv\\_topic\\_attr\\_str\\_setopt](#) (lbm\_rcv\_topic\_attr\_t \*attr, const char \*optname, const char \*optval)  
*Set an option value within the given receiver topic attribute.*
- LBMEExpDLL int [lbm\\_rcv\\_topic\\_attr\\_getopt](#) (lbm\_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 [lbm\\_rcv\\_topic\\_attr\\_str\\_getopt](#) (lbm\_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 [lbm\\_src\\_channel\\_create](#) (lbm\_src\_channel\_info\_t \*\*chnp, lbm\_src\_t \*src, lbm\_uint32\_t channel\_num)  
*Create a channel info object to send messages with the given channel\_num.*
- LBMEExpDLL int [lbm\\_src\\_channel\\_delete](#) (lbm\_src\_channel\_info\_t \*chn)  
*Release the resources associated with a source channel.*
- LBMEExpDLL int [lbm\\_src\\_create](#) (lbm\_src\_t \*\*srp, lbm\_context\_t \*ctx, lbm\_topic\_t \*topic, [lbm\\_src\\_cb\\_proc](#) proc, void \*clientd, lbm\_event\_queue\_t \*evq)

*Create a UM source that will send messages to the given topic.*

- LBMEExpDLL int `lbm_event_queue_dump` (`lbm_event_queue_t *evq`, int \*size, `lbm_config_option_t *opts`)

*Retrieves all event queue attribute options.*

- LBMEExpDLL int `lbm_event_queue_attr_dump` (`lbm_event_queue_attr_t *eattr`, int \*size, `lbm_config_option_t *opts`)

*Retrieves all event queue attribute options.*

- LBMEExpDLL int `lbm_event_queue_attr_option_size` ()

*Retrieves the number of options that are of type "event queue".*

- LBMEExpDLL int `lbm_rcv_create` (`lbm_rcv_t **rcvp`, `lbm_context_t *ctx`, `lbm_topic_t *topic`, `lbm_rcv_cb_proc` proc, void \*clientd, `lbm_event_queue_t *evq`)

*Create a UM receiver that will receive messages sent to the given topic.*

- LBMEExpDLL int `lbm_rcv_subscribe_channel` (`lbm_rcv_t *rcv`, `lbm_uint32_t` channel, `lbm_rcv_cb_proc` proc, void \*clientd)

*Subscribe to a channel, with an optional callback and clientd data pointer.*

- LBMEExpDLL int `lbm_rcv_unsubscribe_channel` (`lbm_rcv_t *rcv`, `lbm_uint32_t` channel)

*Discontinue an existing channel subscription.*

- LBMEExpDLL int `lbm_rcv_unsubscribe_channel_ex` (`lbm_rcv_t *rcv`, `lbm_uint32_t` channel, `lbm_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 `lbm_wildcard_rcv_subscribe_channel` (`lbm_wildcard_rcv_t *wrcv`, `lbm_uint32_t` channel, `lbm_rcv_cb_proc` proc, void \*clientd)

*Subscribe to a channel, with an optional callback and clientd data pointer.*

- LBMEExpDLL int `lbm_wildcard_rcv_unsubscribe_channel` (`lbm_wildcard_rcv_t *wrcv`, `lbm_uint32_t` channel)

*Discontinue an existing channel subscription.*

- LBMEExpDLL int `lbm_wildcard_rcv_unsubscribe_channel_ex` (`lbm_wildcard_rcv_t *wrcv`, `lbm_uint32_t` channel, `lbm_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.

- LBMEpDLL int `lbm_src_delete` (`lbm_src_t *src`)  
*Delete a UM source object.*
- LBMEpDLL int `lbm_src_delete_ex` (`lbm_src_t *src`, `lbm_event_queue_cancel_cb_info_t *cbinfo`)  
*Extended delete a UM source object.*
- LBMEpDLL `lbm_context_t *lbm_context_from_src` (`lbm_src_t *src`)  
*Retrieve the UM context object associated with a UM source object.*
- LBMEpDLL `lbm_topic_t *lbm_topic_from_src` (`lbm_src_t *src`)  
*Retrieve the UM topic object associated with a UM source object.*
- LBMEpDLL `lbm_event_queue_t *lbm_event_queue_from_src` (`lbm_src_t *src`)  
*Retrieve the UM event queue object associated with a UM source object.*
- LBMEpDLL int `lbm_rcv_delete` (`lbm_rcv_t *rcv`)  
*Delete a UM receiver object.*
- LBMEpDLL int `lbm_rcv_delete_ex` (`lbm_rcv_t *rcv`, `lbm_event_queue_cancel_cb_info_t *cbinfo`)  
*Extended delete a UM receiver object.*
- LBMEpDLL `lbm_context_t *lbm_context_from_rcv` (`lbm_rcv_t *rcv`)  
*Retrieve the UM context object associated with a UM receiver object.*
- LBMEpDLL `lbm_event_queue_t *lbm_event_queue_from_rcv` (`lbm_rcv_t *rcv`)  
*Retrieve the UM event queue object associated with a UM receiver object.*
- LBMEpDLL int `lbm_src_setopt` (`lbm_src_t *src`, `const char *optname`, `const void *optval`, `size_t optlen`)  
*Set an option value within the given src.*
- LBMEpDLL int `lbm_src_str_setopt` (`lbm_src_t *src`, `const char *optname`, `const char *optval`)  
*Set an option value within the given src.*

- LBMEExpDLL int [lbm\\_src\\_getopt](#) (lbm\_src\_t \*src, const char \*optname, void \*optval, size\_t \*optlen)  
*Retrieve an option value within the given src.*
- LBMEExpDLL int [lbm\\_src\\_str\\_getopt](#) (lbm\_src\_t \*src, const char \*optname, char \*optval, size\_t \*optlen)  
*Retrieve a textual option value within the given src.*
- LBMEExpDLL int [lbm\\_rcv\\_setopt](#) (lbm\_rcv\_t \*rcv, const char \*optname, const void \*optval, size\_t optlen)  
*Set an option value within the given rcv.*
- LBMEExpDLL int [lbm\\_rcv\\_str\\_setopt](#) (lbm\_rcv\_t \*rcv, const char \*optname, const char \*optval)  
*Set an option value within the given rcv.*
- LBMEExpDLL int [lbm\\_rcv\\_getopt](#) (lbm\_rcv\_t \*rcv, const char \*optname, void \*optval, size\_t \*optlen)  
*Retrieve an option value within the given rcv.*
- LBMEExpDLL int [lbm\\_rcv\\_str\\_getopt](#) (lbm\_rcv\_t \*rcv, const char \*optname, char \*optval, size\_t \*optlen)  
*Retrieve a textual option value within the given rcv.*
- LBMEExpDLL int [lbm\\_src\\_send](#) (lbm\_src\_t \*src, const char \*msg, size\_t len, int flags)  
*Send a message to the topic associated with a UM source.*
- LBMEExpDLL int [lbm\\_src\\_send\\_ex](#) (lbm\_src\_t \*src, 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 a UM source.*
- LBMEExpDLL int [lbm\\_src\\_flush](#) (lbm\_src\_t \*src)  
*Send messages from both the explicit and implicit batches ASAP.*
- LBMEExpDLL int [lbm\\_src\\_sendv](#) (lbm\_src\_t \*src, const [lbm\\_iovec\\_t](#) \*iov, int num, int flags)  
*Send a set of messages to the topic associated with a UM source.*
- LBMEExpDLL 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)  
*Extended send of a set of messages to the topic associated with a UM source.*

- LBMEpDLL int [lbm\\_rcv\\_msg\\_source\\_clientd](#) (lbm\_rcv\_t \*rcv, const char \*source, void \*source\_clientd)  
*Set the pointer value to set in the messages received for the given receiver from a specific source.*
- LBMEpDLL int [lbm\\_src\\_retrieve\\_transport\\_stats](#) (lbm\_src\_t \*src, [lbm\\_src\\_transport\\_stats\\_t](#) \*stats)  
*Retrieve the transport statistics for the transport used by the given source.*
- LBMEpDLL int [lbm\\_src\\_reset\\_transport\\_stats](#) (lbm\_src\_t \*src)  
*Reset the transport statistics for the transport used by the given source.*
- LBMEpDLL int [lbm\\_rcv\\_retrieve\\_transport\\_stats](#) (lbm\_rcv\_t \*rcv, const char \*source, [lbm\\_rcv\\_transport\\_stats\\_t](#) \*stats)  
*Retrieve the transport statistics for the transport used by the given receiver from a specific source.*
- LBMEpDLL int [lbm\\_rcv\\_reset\\_transport\\_stats](#) (lbm\_rcv\_t \*rcv, const char \*source)  
*Reset the transport statistics for the transport used by the given receiver from a specific source.*
- LBMEpDLL int [lbm\\_rcv\\_retrieve\\_all\\_transport\\_stats](#) (lbm\_rcv\_t \*rcv, int \*num, [lbm\\_rcv\\_transport\\_stats\\_t](#) \*stats)  
*Retrieve the transport stats for all the sources seen by the given receiver.*
- LBMEpDLL int [lbm\\_rcv\\_retrieve\\_all\\_transport\\_stats\\_ex](#) (lbm\_rcv\_t \*rcv, int \*num, int size, [lbm\\_rcv\\_transport\\_stats\\_t](#) \*stats)
- LBMEpDLL int [lbm\\_rcv\\_reset\\_all\\_transport\\_stats](#) (lbm\_rcv\_t \*rcv)  
*Reset the transport stats for all the sources seen by the given receiver.*
- LBMEpDLL int [lbm\\_context\\_retrieve\\_rcv\\_transport\\_stats](#) (lbm\_context\_t \*ctx, int \*num, [lbm\\_rcv\\_transport\\_stats\\_t](#) \*stats)  
*Retrieve the transport stats for all receivers in a given context.*
- LBMEpDLL int [lbm\\_context\\_retrieve\\_rcv\\_transport\\_stats\\_ex](#) (lbm\_context\_t \*ctx, int \*num, int size, [lbm\\_rcv\\_transport\\_stats\\_t](#) \*stats)
- LBMEpDLL int [lbm\\_context\\_reset\\_rcv\\_transport\\_stats](#) (lbm\_context\_t \*ctx)  
*Reset the transport stats for all receivers in a given context.*
- LBMEpDLL int [lbm\\_context\\_retrieve\\_src\\_transport\\_stats](#) (lbm\_context\_t \*ctx, int \*num, [lbm\\_src\\_transport\\_stats\\_t](#) \*stats)

*Retrieve the transport stats for all the sources in a given context.*

- LBMEExpDLL int **lbm\_context\_retrieve\_src\_transport\_stats\_ex** (lbm\_context\_t \*ctx, int \*num, int size, **lbm\_src\_transport\_stats\_t** \*stats)
- LBMEExpDLL int **lbm\_context\_reset\_src\_transport\_stats** (lbm\_context\_t \*ctx)

*Reset the transport stats for all the sources in a given context.*

- LBMEExpDLL int **lbm\_event\_queue\_retrieve\_stats** (lbm\_event\_queue\_t \*evq, **lbm\_event\_queue\_stats\_t** \*stats)

*Retrieve the stats for an event queue.*

- LBMEExpDLL int **lbm\_event\_queue\_reset\_stats** (lbm\_event\_queue\_t \*evq)

*Reset the stats for an event queue.*

- LBMEExpDLL int **lbm\_context\_retrieve\_stats** (lbm\_context\_t \*ctx, **lbm\_context\_stats\_t** \*stats)

*Retrieve the stats for a context.*

- LBMEExpDLL int **lbm\_context\_reset\_stats** (lbm\_context\_t \*ctx)

*Reset the stats for a context.*

- LBMEExpDLL int **lbm\_context\_retrieve\_im\_src\_transport\_stats** (lbm\_context\_t \*ctx, int \*num, int size, **lbm\_src\_transport\_stats\_t** \*stats)

*Retrieve the IM source stats for a context.*

- LBMEExpDLL int **lbm\_context\_reset\_im\_src\_transport\_stats** (lbm\_context\_t \*ctx)

*Reset the IM source stats for a context.*

- LBMEExpDLL int **lbm\_context\_retrieve\_im\_rcv\_transport\_stats** (lbm\_context\_t \*ctx, int \*num, int size, **lbm\_rcv\_transport\_stats\_t** \*stats)

*Retrieve the IM receiver stats for a context.*

- LBMEExpDLL int **lbm\_context\_reset\_im\_rcv\_transport\_stats** (lbm\_context\_t \*ctx)

*Reset the IM receiver stats for a context.*

- LBMEExpDLL int **lbm\_msg\_retain** (lbm\_msg\_t \*msg)

*Instruct UM that the API is going to retain ownership of a UM message object.*

- LBMEExpDLL int **lbm\_msg\_is\_fragment** (lbm\_msg\_t \*msg)

*Retrieve fragment information from a UM message.*

- LBMEpDLL int `lbm_msg_retrieve_fragment_info` (`lbm_msg_t *msg`, `lbm_msg_fragment_info_t *info`)  
*Returns 1 if lbm message is a fragment, else 0 is returned.*
- LBMEpDLL int `lbm_msg_retrieve_gateway_info` (`lbm_msg_t *msg`, `lbm_msg_gateway_info_t *info`)  
*Retrieve gateway information from a UM message.*
- LBMEpDLL int `lbm_msg_retrieve_msgid` (`lbm_msg_t *msg`, `lbm_umq_msgid_t *id`)  
*Retrieve UMQ Message ID information from a UM message.*
- LBMEpDLL int `lbm_msg_retrieve_umq_index` (`lbm_msg_t *msg`, `lbm_umq_index_info_t *info`)  
*Retrieve UMQ index information from a UM message.*
- LBMEpDLL int `lbm_msg_retrieve_delivery_latency` (`lbm_msg_t *msg`, `lbm_int64_t *latency_nsecs`)
- LBMEpDLL int `lbm_msg_delete` (`lbm_msg_t *msg`)  
*Delete a UM message object.*
- LBMEpDLL int `lbm_msg_ume_send_explicit_ack` (`lbm_msg_t *msg`)  
*Send an Explicit UMP ACK for a UM message object.*
- LBMEpDLL int `lbm_msg_ume_can_send_explicit_ack` (`lbm_msg_t *msg`)  
*Check to see if Explicit UMP ACK for a UM message object can be called.*
- LBMEpDLL int `lbm_src_ume_deregister` (`lbm_src_t *src`)  
*Deregister a source from the UMP stores.*
- LBMEpDLL int `lbm_rcv_ume_deregister` (`lbm_rcv_t *rcv`)  
*Deregister a receiver from all known UMP stores.*
- LBMEpDLL int `lbm_wrcv_ume_deregister` (`lbm_wildcard_rcv_t *wrcv`)  
*Deregister a wildcard receiver from all known UMP stores.*
- LBMEpDLL int `lbm_msg_umq_reassign` (`lbm_msg_t *msg`, int flags)  
*Do not acknowledge the given message and instead request that the message be re-assigned.*
- LBMEpDLL int `lbm_rcv_umq_deregister` (`lbm_rcv_t *rcv`, const char \*queue\_name)  
*De-Register the given receiver from the given UMQ queue or all UMQ queues.*

- LBMEpDLL int [lbm\\_rcv\\_umq\\_index\\_stop\\_assignment](#) (lbm\_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.*
- LBMEpDLL int [lbm\\_rcv\\_umq\\_index\\_start\\_assignment](#) (lbm\_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.*
- LBMEpDLL int [lbm\\_rcv\\_umq\\_index\\_reserve](#) (lbm\_rcv\_t \*rcv, const char \*queue\_name, lbm\_umq\_index\_info\_t \*index\_info)  
*Instruct the given UMQ queue(s) to reserve an index for assignment to this receiver.*
- LBMEpDLL int [lbm\\_rcv\\_umq\\_index\\_release](#) (lbm\_rcv\_t \*rcv, const char \*queue\_name, lbm\_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.*
- LBMEpDLL int [lbm\\_wildcard\\_rcv\\_umq\\_index\\_stop\\_assignment](#) (lbm\_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.*
- LBMEpDLL int [lbm\\_wildcard\\_rcv\\_umq\\_index\\_start\\_assignment](#) (lbm\_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.*
- LBMEpDLL int [lbm\\_wildcard\\_rcv\\_umq\\_index\\_release](#) (lbm\_wildcard\_rcv\_t \*wrcv, const char \*queue\_name, lbm\_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.*
- LBMEpDLL int [lbm\\_wildcard\\_rcv\\_umq\\_deregister](#) (lbm\_wildcard\_rcv\_t \*wrcv, const char \*queue\_name)  
*De-Register the given wildcard receiver from the given UMQ queue or all UMQ queues.*
- LBMEpDLL int [lbm\\_send\\_response](#) (lbm\_response\_t \*resp, const char \*data, size\_t len, int flags)  
*Send a response for a given resp response.*

- LBMEExpDLL int [lbm\\_response\\_delete](#) (lbm\_response\_t \*resp)  
*Delete a UM response object.*
- LBMEExpDLL int [lbm\\_serialized\\_response\\_delete](#) (lbm\_serialized\_response\_t \*serialized\_response)  
*Delete a UM serialized response object.*
- LBMEExpDLL [lbm\\_serialized\\_response\\_t](#) \* [lbm\\_serialize\\_response](#) (lbm\_response\_t \*resp)  
*Serialize a UM response object.*
- LBMEExpDLL lbm\_response\_t \* [lbm\\_deserialize\\_response](#) (lbm\_context\_t \*ctx, lbm\_serialized\_response\_t \*serialized\_response)  
*De-serialize a UM response object.*
- LBMEExpDLL int [lbm\\_send\\_request](#) (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)  
*Send a request on the given src that contains the given data.*
- LBMEExpDLL 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)  
*Send a request on the given src that contains the given data.*
- LBMEExpDLL int [lbm\\_request\\_delete](#) (lbm\_request\_t \*req)  
*Delete a UM request object.*
- LBMEExpDLL int [lbm\\_request\\_delete\\_ex](#) (lbm\_request\_t \*req, [lbm\\_event\\_queue\\_cancel\\_cb\\_info\\_t](#) \*cbinfo)  
*Extended delete a UM request object.*
- 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)  
*Create a UM event queue object.*
- LBMEExpDLL int [lbm\\_event\\_queue\\_attr\\_create](#) (lbm\_event\_queue\_attr\_t \*\*attr)  
*Create and fill a UM event queue attribute object with the current default values.*
- LBMEExpDLL int [lbm\\_event\\_queue\\_attr\\_create\\_default](#) (lbm\_event\_queue\_attr\_t \*\*attr)

*Create and fill a UM event queue attribute object with the initial default values.*

- LBMEExpDLL int `lbm_event_queue_attr_create_from_xml` (lbm\_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 `lbm_event_queue_attr_set_from_xml` (lbm\_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 `lbm_event_queue_attr_delete` (lbm\_event\_queue\_attr\_t \*attr)

*Delete an event queue attribute object.*

- LBMEExpDLL int `lbm_event_queue_attr_dup` (lbm\_event\_queue\_attr\_t \*\*attr, const lbm\_event\_queue\_attr\_t \*original)

*Duplicate a UM event queue attribute object.*

- LBMEExpDLL int `lbm_event_queue_attr_setopt` (lbm\_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 `lbm_event_queue_attr_str_setopt` (lbm\_event\_queue\_attr\_t \*attr, const char \*optname, const char \*optval)

*Set an option value within the given event queue attribute.*

- LBMEExpDLL int `lbm_event_queue_attr_getopt` (lbm\_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 `lbm_event_queue_attr_str_getopt` (lbm\_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 `lbm_event_queue_setopt` (lbm\_event\_queue\_t \*evq, const char \*optname, const void \*optval, size\_t optlen)

*Set an option value within the given evq.*

- LBMEExpDLL int `lbm_event_queue_str_setopt` (lbm\_event\_queue\_t \*evq, const char \*optname, const char \*optval)

*Set an option value within the given event queue.*

- LBMEExpDLL int [lbm\\_event\\_queue\\_getopt](#) (lbm\_event\_queue\_t \*evq, const char \*optname, void \*optval, size\_t \*optlen)  
*Retrieve an option value within the given event queue.*
- LBMEExpDLL int [lbm\\_event\\_queue\\_str\\_getopt](#) (lbm\_event\_queue\_t \*evq, const char \*optname, char \*optval, size\_t \*optlen)  
*Retrieve a textual option value within the given event queue.*
- LBMEExpDLL int [lbm\\_event\\_dispatch](#) (lbm\_event\_queue\_t \*evq, lbm\_ulong\_t tmo)  
*Dispatch waiting events to appropriate callback functions.*
- LBMEExpDLL int [lbm\\_event\\_dispatch\\_unblock](#) (lbm\_event\_queue\_t \*evq)  
*Unblock the given UM event queue object so that a thread waiting in `lbm_event_dispatch` returns as soon as feasible.*
- LBMEExpDLL int [lbm\\_event\\_queue\\_size](#) (lbm\_event\_queue\_t \*evq)  
*Determine the number of queued events in the event queue.*
- LBMEExpDLL int [lbm\\_event\\_queue\\_shutdown](#) (lbm\_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 [lbm\\_event\\_queue\\_delete](#) (lbm\_event\_queue\_t \*evq)  
*Delete a given UM event queue object.*
- LBMEExpDLL int [lbm\\_unicast\\_immediate\\_message](#) (lbm\_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 [lbm\\_unicast\\_immediate\\_request](#) (lbm\_request\_t \*\*reqp, lbm\_context\_t \*ctx, const char \*target, const char \*topic, const char \*data, size\_t len, [lbm\\_request\\_cb\\_proc](#) proc, void \*clientd, lbm\_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 [lbm\\_queue\\_immediate\\_message](#) (lbm\_context\_t \*ctx, const char \*qname, const char \*topic, const char \*data, size\_t len, int flags, [lbm\\_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 [lbm\\_multicast\\_immediate\\_message](#) (lbm\_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 [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)

*Multicast an immediate request to the target and topic. Note that immediate messages are processed somewhat less efficiently than source-based messages.*
- LBMEExpDLL int [lbm\\_wildcard\\_rcv\\_dump](#) (lbm\_wildcard\_rcv\_t \*wrcv, int \*size, lbm\_config\_option\_t \*opts)

*Retrieves all wildcard receiver attribute options.*
- LBMEExpDLL int [lbm\\_wildcard\\_rcv\\_attr\\_dump](#) (lbm\_wildcard\_rcv\_attr\_t \*wattr, int \*size, lbm\_config\_option\_t \*opts)

*Retrieves all wildcard receiver attribute options.*
- LBMEExpDLL int [lbm\\_wildcard\\_rcv\\_attr\\_option\\_size](#) ()

*Retrieves the number of options that are of type "wildcard receiver".*
- LBMEExpDLL int [lbm\\_wildcard\\_rcv\\_attr\\_create](#) (lbm\_wildcard\_rcv\_attr\_t \*\*attr)

*Create and fill a UM wildcard receiver attribute object with the current default values.*
- LBMEExpDLL int [lbm\\_wildcard\\_rcv\\_attr\\_create\\_default](#) (lbm\_wildcard\_rcv\_attr\_t \*\*attr)

*Create and fill a UM wildcard receiver attribute object with the initial default values.*
- LBMEExpDLL int [lbm\\_wildcard\\_rcv\\_attr\\_create\\_from\\_xml](#) (lbm\_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 [lbm\\_wildcard\\_rcv\\_attr\\_set\\_from\\_xml](#) (lbm\_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 [lbm\\_wildcard\\_rcv\\_attr\\_delete](#) (lbm\_wildcard\_rcv\_attr\_t \*attr)

*Delete a wildcard receiver attribute object.*

- LBMEpDLL int [lbm\\_wildcard\\_rcv\\_attr\\_dup](#) (lbm\_wildcard\_rcv\_attr\_t \*\*attr, const lbm\_wildcard\_rcv\_attr\_t \*original)  
*Duplicate a UM wildcard receiver attribute object.*
- LBMEpDLL int [lbm\\_wildcard\\_rcv\\_attr\\_setopt](#) (lbm\_wildcard\_rcv\_attr\_t \*attr, const char \*optname, const void \*optval, size\_t optlen)  
*Set an option value within the given wildcard receiver attribute.*
- LBMEpDLL int [lbm\\_wildcard\\_rcv\\_attr\\_str\\_setopt](#) (lbm\_wildcard\_rcv\_attr\_t \*attr, const char \*optname, const char \*optval)  
*Set an option value within the given wildcard receiver attribute.*
- LBMEpDLL int [lbm\\_wildcard\\_rcv\\_attr\\_getopt](#) (lbm\_wildcard\_rcv\_attr\_t \*attr, const char \*optname, void \*optval, size\_t \*optlen)  
*Retrieve an option value within the given wildcard receiver attribute.*
- LBMEpDLL int [lbm\\_wildcard\\_rcv\\_attr\\_str\\_getopt](#) (lbm\_wildcard\_rcv\_attr\_t \*attr, const char \*optname, char \*optval, size\_t \*optlen)  
*Retrieve a textual option value within the given wildcard receiver attribute.*
- LBMEpDLL int [lbm\\_wildcard\\_rcv\\_setopt](#) (lbm\_wildcard\_rcv\_t \*wrcv, const char \*optname, const void \*optval, size\_t optlen)  
*Set an option value within the given wrcv.*
- LBMEpDLL int [lbm\\_wildcard\\_rcv\\_str\\_setopt](#) (lbm\_wildcard\_rcv\_t \*wrcv, const char \*optname, const char \*optval)  
*Set an option value within the given wrcv.*
- LBMEpDLL int [lbm\\_wildcard\\_rcv\\_getopt](#) (lbm\_wildcard\_rcv\_t \*wrcv, const char \*optname, void \*optval, size\_t \*optlen)  
*Retrieve an option value within the given wrcv.*
- LBMEpDLL int [lbm\\_wildcard\\_rcv\\_str\\_getopt](#) (lbm\_wildcard\_rcv\_t \*wrcv, const char \*optname, char \*optval, size\_t \*optlen)  
*Retrieve the textual option value within the given wrcv.*
- LBMEpDLL int [lbm\\_wildcard\\_rcv\\_create](#) (lbm\_wildcard\_rcv\_t \*\*wrcvp, lbm\_context\_t \*ctx, const char \*pattern, const lbm\_rcv\_topic\_attr\_t \*tattr, const lbm\_wildcard\_rcv\_attr\_t \*wattr, [lbm\\_rcv\\_cb\\_proc](#) proc, void \*clientd, lbm\_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).*

- LBMEpDLL int [lbm\\_wildcard\\_rcv\\_delete](#) (lbm\_wildcard\_rcv\_t \*wrcv)  
*Delete a UM wildcard receiver object.*
- LBMEpDLL int [lbm\\_wildcard\\_rcv\\_delete\\_ex](#) (lbm\_wildcard\_rcv\_t \*wrcv, [lbm\\_event\\_queue\\_cancel\\_cb\\_info\\_t](#) \*cbinfo)  
*Extended delete a UM wildcard receiver object.*
- LBMEpDLL [lbm\\_context\\_t](#) \* [lbm\\_context\\_from\\_wildcard\\_rcv](#) (lbm\_wildcard\_rcv\_t \*wrcv)  
*Retrieve the LBM context object associated with a UM wildcard receiver object.*
- LBMEpDLL [lbm\\_event\\_queue\\_t](#) \* [lbm\\_event\\_queue\\_from\\_wildcard\\_rcv](#) (lbm\_wildcard\_rcv\_t \*wrcv)  
*Retrieve the LBM event queue object associated with a UM wildcard receiver object.*
- LBMEpDLL int [lbm\\_hf\\_src\\_create](#) (lbm\_src\_t \*\*srpc, [lbm\\_context\\_t](#) \*ctx, [lbm\\_topic\\_t](#) \*topic, [lbm\\_src\\_cb\\_proc](#) proc, void \*clientd, [lbm\\_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.*
- LBMEpDLL int [lbm\\_hf\\_src\\_send](#) (lbm\_src\_t \*src, const char \*msg, size\_t len, [lbm\\_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.*
- LBMEpDLL 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)  
*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.*
- LBMEpDLL int [lbm\\_hf\\_src\\_sendv](#) (lbm\_src\_t \*src, const [lbm\\_iovec\\_t](#) \*iov, int num, [lbm\\_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.*
- LBMEpDLL int [lbm\\_hf\\_rcv\\_topic\\_dump](#) ([lbm\\_hf\\_rcv\\_t](#) \*hfrcv, int \*size, [lbm\\_config\\_option\\_t](#) \*opts)  
*Retrieves all receiver attribute options for an HF receiver.*

- LBMEpDLL int [lbm\\_hf\\_src\\_sendv\\_ex](#) (lbm\_src\_t \*src, const [lbm\\_iovec\\_t](#) \*iov, int num, lbm\_uint\_t sqn, int flags, [lbm\\_src\\_send\\_ex\\_info\\_t](#) \*exinfo)

*Extended send of a set of Hot Failover (HF) messages to the topic associated with a UM source.*
- LBMEpDLL int [lbm\\_hf\\_src\\_send\\_rcv\\_reset](#) (lbm\_src\_t \*src, int flags, [lbm\\_src\\_send\\_ex\\_info\\_t](#) \*exinfo)

*Send a message that will reset order and loss information for hot failover receivers on this topic.*
- LBMEpDLL int [lbm\\_hf\\_rcv\\_create](#) (lbm\_hf\_rcv\_t \*\*hfrcvp, lbm\_context\_t \*ctx, lbm\_topic\_t \*topic, [lbm\\_rcv\\_cb\\_proc](#) proc, void \*clientd, lbm\_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.*
- LBMEpDLL int [lbm\\_hf\\_rcv\\_delete](#) (lbm\_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.*
- LBMEpDLL int [lbm\\_hf\\_rcv\\_delete\\_ex](#) (lbm\_hf\_rcv\_t \*hfrcv, [lbm\\_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.*
- LBMEpDLL lbm\_hf\_rcv\_t \* [lbm\\_hf\\_rcv\\_from\\_rcv](#) (lbm\_rcv\_t \*rcv)

*Return the LBM Hot Failover (HF) receiver object (if any) from a UM receiver object.*
- LBMEpDLL lbm\_rcv\_t \* [lbm\\_rcv\\_from\\_hf\\_rcv](#) (lbm\_hf\_rcv\_t \*hfrcv)

*Return the LBM receiver object associated with a UM Hot Failover (HF) receiver object.*
- LBMEpDLL int [lbm\\_hfx\\_dump](#) (lbm\_hfx\_t \*hfx, int \*size, lbm\_config\_option\_t \*opts)

*Retrieves all HFX attribute options.*
- LBMEpDLL int [lbm\\_hfx\\_attr\\_dump](#) (lbm\_hfx\_attr\_t \*attr, int \*size, lbm\_config\_option\_t \*opts)

*Retrieves all HFX attribute options.*

- LBMEExpDLL int [lbm\\_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 [lbm\\_hfx\\_attr\\_create](#) (lbm\_hfx\_attr\_t \*\*attr)  
*Create and fill a UM HFX attribute object with the current default values.*
- LBMEExpDLL int [lbm\\_hfx\\_attr\\_create\\_default](#) (lbm\_hfx\_attr\_t \*\*attr)  
*Create and fill a UM HFX attribute object with the initial default values.*
- LBMEExpDLL int [lbm\\_hfx\\_attr\\_create\\_from\\_xml](#) (lbm\_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 [lbm\\_hfx\\_attr\\_set\\_from\\_xml](#) (lbm\_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 [lbm\\_hfx\\_attr\\_delete](#) (lbm\_hfx\_attr\_t \*attr)  
*Delete a UM hfx attribute object.*
- LBMEExpDLL int [lbm\\_hfx\\_attr\\_dup](#) (lbm\_hfx\_attr\_t \*\*attr, const lbm\_hfx\_attr\_t \*original)  
*Duplicate a UM hfx attribute object.*
- LBMEExpDLL int [lbm\\_hfx\\_attr\\_setopt](#) (lbm\_hfx\_attr\_t \*attr, const char \*optname, const void \*optval, size\_t optlen)  
*Set an option for the given LBM hfx attribute.*
- LBMEExpDLL int [lbm\\_hfx\\_attr\\_str\\_setopt](#) (lbm\_hfx\_attr\_t \*attr, const char \*optname, const char \*optval)  
*Set an option for the given LBM hfx attribute using a string.*
- LBMEExpDLL int [lbm\\_hfx\\_attr\\_getopt](#) (lbm\_hfx\_attr\_t \*attr, const char \*optname, void \*optval, size\_t \*optlen)  
*Retrieve the value of an option for the given LBM hfx attribute.*
- LBMEExpDLL int [lbm\\_hfx\\_attr\\_str\\_getopt](#) (lbm\_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.*

- LBMEExpDLL int [lbm\\_hfx\\_setopt](#) (lbm\_hfx\_t \*hfx, const char \*optname, const void \*optval, size\_t optlen)  
*Set an option value within the given hfx.*
- LBMEExpDLL int [lbm\\_hfx\\_str\\_setopt](#) (lbm\_hfx\_t \*hfx, const char \*optname, const char \*optval)  
*Set an option value within the given hfx.*
- LBMEExpDLL int [lbm\\_hfx\\_getopt](#) (lbm\_hfx\_t \*hfx, const char \*optname, void \*optval, size\_t \*optlen)  
*Retrieve an option value within the given hfx.*
- LBMEExpDLL int [lbm\\_hfx\\_str\\_getopt](#) (lbm\_hfx\_t \*hfx, const char \*optname, char \*optval, size\_t \*optlen)  
*Retrieve the textual option value within the given hfx.*
- LBMEExpDLL int [lbm\\_hfx\\_create](#) (lbm\_hfx\_t \*\*hfxp, lbm\_hfx\_attr\_t \*cattr, const char \*symbol, [lbm\\_rcv\\_cb\\_proc](#) proc, lbm\_event\_queue\_t \*evq)  
*Create and initialize an lbm\_hfx\_t object.*
- LBMEExpDLL int [lbm\\_hfx\\_delete](#) (lbm\_hfx\_t \*hfx)  
*Delete a UM hfx object.*
- LBMEExpDLL int [lbm\\_hfx\\_delete\\_ex](#) (lbm\_hfx\_t \*hfx, [lbm\\_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 [lbm\\_hfx\\_rcv\\_topic\\_dump](#) (lbm\_hfx\_rcv\_t \*hfxrcv, int \*size, lbm\_config\_option\_t \*opts)  
*Retrieves all receiver attribute options for an HFX receiver.*
- LBMEExpDLL int [lbm\\_hfx\\_rcv\\_create](#) (lbm\_hfx\_rcv\_t \*\*hfxrcvp, lbm\_hfx\_t \*hfx, lbm\_context\_t \*ctx, lbm\_rcv\_topic\_attr\_t \*rattr, void \*clientd)  
*Create a HFX receiver.*
- LBMEExpDLL lbm\_rcv\_t \* [lbm\\_rcv\\_from\\_hfx\\_rcv](#) (lbm\_hfx\_rcv\_t \*hfxrcv)  
*Retrieve the underlying receiver from an lbm\_hfx\_rcv\_t.*
- LBMEExpDLL int [lbm\\_hfx\\_rcv\\_delete](#) (lbm\_hfx\_rcv\_t \*hfxrcv)  
*Delete a HFX receiver.*

- LBMEpDLL int [lbm\\_hfx\\_rcv\\_delete\\_ex](#) (lbm\_hfx\_rcv\_t \*hfrcv, [lbm\\_event\\_queue\\_cancel\\_cb\\_info\\_t](#) \*cbinfo)  
*Extended delete a UM HFX receiver object.*
- LBMEpDLL void [lbm\\_debug\\_filename](#) (const char \*filename)  
*Set the file to receive LBM debug log entries.*
- LBMEpDLL void [lbm\\_debug\\_mask](#) (lbm\_uint64\_t mask)  
*Set the debug mask for LBM debug log entries.*
- LBMEpDLL void [lbm\\_debug\\_noflush](#) (int on)  
*Set the noflush flag for the debug logs. This can dramatically increase performance when a debug mask is set.*
- LBMEpDLL void [lbm\\_log\\_debug](#) (int on)  
*Enable logging of debug messages to the application logging callback set by [lbm\\_log\(\)](#). By default, if lbm debug logging is enabled it is sent to the filename specified by [lbm\\_debug\\_filename\(\)](#), or stderr. Calling [lbm\\_log\\_debug\(\)](#) with a value of 1 will redirect debug logging to the application logging callback set by [lbm\\_log\(\)](#), or stderr if no callback is set.*
- LBMEpDLL int [lbm\\_debug\\_dump](#) (const char \*filename, int append)  
*Dump a running rollback debug log to the given filename.*
- LBMEpDLL void [lbm\\_set\\_uim\\_loss\\_rate](#) (int rate)  
*Dynamically set the UIM loss rate.*
- LBMEpDLL void [lbm\\_set\\_lbtrm\\_loss\\_rate](#) (int rate)  
*Dynamically set the LBT-RM loss rate.*
- LBMEpDLL void [lbm\\_set\\_lbtrm\\_src\\_loss\\_rate](#) (int rate)  
*Dynamically set the LBT-RM source loss rate.*
- LBMEpDLL void [lbm\\_set\\_lbtru\\_loss\\_rate](#) (int rate)  
*Dynamically set the LBT-RU loss rate.*
- LBMEpDLL void [lbm\\_set\\_lbtru\\_src\\_loss\\_rate](#) (int rate)  
*Dynamically set the LBT-RU source loss rate.*
- LBMEpDLL int [lbm\\_transport\\_source\\_parse](#) (const char \*source, [lbm\\_transport\\_source\\_info\\_t](#) \*info, size\_t infosize)  
*Parse a UM transport source string into its components.*

- LBMEExpDLL int [lbm\\_transport\\_source\\_format](#) (const [lbm\\_transport\\_source\\_info\\_t](#) \*info, size\_t infosize, char \*source, size\_t \*size)  
*Format a UM transport source string from its components.*
- LBMEExpDLL int [lbm\\_apphdr\\_chain\\_create](#) (lbm\_apphdr\_chain\_t \*\*chain)  
*Create a new app header chain that can be used to include metadata with a message.*
- LBMEExpDLL int [lbm\\_apphdr\\_chain\\_delete](#) (lbm\_apphdr\_chain\_t \*chain)  
*Delete an app header chain previously created with [lbm\\_apphdr\\_chain\\_create](#).*
- LBMEExpDLL int [lbm\\_apphdr\\_chain\\_append\\_elem](#) (lbm\_apphdr\_chain\_t \*chain, [lbm\\_apphdr\\_chain\\_elem\\_t](#) \*elem)  
*Appends a user-created app header to an app header chain.*
- LBMEExpDLL int [lbm\\_apphdr\\_chain\\_iter\\_create](#) (lbm\_apphdr\_chain\_iter\_t \*\*chain\_iter, lbm\_apphdr\_chain\_t \*chain)  
*Create an iterator (an [lbm\\_apphdr\\_chain\\_iter\\_t](#) structure) to point to the first element in an apphdr chain.*
- LBMEExpDLL int [lbm\\_apphdr\\_chain\\_iter\\_create\\_from\\_msg](#) (lbm\_apphdr\_chain\_iter\_t \*\*chain\_iter, [lbm\\_msg\\_t](#) \*msg)  
*Create an iterator (an [lbm\\_apphdr\\_chain\\_iter\\_t](#) structure) to point to the first element in an apphdr chain associated with a UM message.*
- LBMEExpDLL int [lbm\\_apphdr\\_chain\\_iter\\_delete](#) (lbm\_apphdr\_chain\_iter\_t \*chain\_iter)  
*Delete an iterator allocated by one of the [lbm\\_apphdr\\_chain\\_iter](#) functions.*
- LBMEExpDLL int [lbm\\_apphdr\\_chain\\_iter\\_first](#) (lbm\_apphdr\_chain\_iter\_t \*\*chain\_iter)  
*Initializes an app header chain iterator to the first element in the chain.*
- LBMEExpDLL int [lbm\\_apphdr\\_chain\\_iter\\_done](#) (lbm\_apphdr\_chain\_iter\_t \*\*chain\_iter)  
*Tests an [lbm\\_apphdr\\_chain\\_iter\\_t](#) iterator to see if more elements in the chain remain.*
- LBMEExpDLL int [lbm\\_apphdr\\_chain\\_iter\\_next](#) (lbm\_apphdr\_chain\_iter\_t \*\*chain\_iter)  
*Advances the iterator to the next element in an app header chain, if any.*
- LBMEExpDLL [lbm\\_apphdr\\_chain\\_elem\\_t](#) \* [lbm\\_apphdr\\_chain\\_iter\\_current](#) (lbm\_apphdr\_chain\_iter\_t \*\*chain\_iter)

Returns the current element of an app header chain pointed to by an `lbm_apphdr_chain_iter_t` iterator.

- LBMEpDLL int `lbm_msg_properties_create` (`lbm_msg_properties_t **properties`)

Creates a new properties object, used for sending messages with properties.

- LBMEpDLL int `lbm_msg_properties_delete` (`lbm_msg_properties_t *properties`)

Deletes a properties object.

- LBMEpDLL int `lbm_msg_properties_set` (`lbm_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.

- LBMEpDLL int `lbm_msg_properties_clear` (`lbm_msg_properties_t *properties`, `const char *name`)

Clear the value associated with the name in the specified properties object.

- LBMEpDLL int `lbm_msg_properties_get` (`lbm_msg_properties_t *properties`, `const char *name`, `void *value`, `int *type`, `size_t *size`)

Gets the value of the property with the specified name.

- LBMEpDLL int `lbm_msg_properties_iter_create` (`lbm_msg_properties_iter_t **iterp`)

Creates a new msg properties iterator. The newly created iterator is not associated with any properties object. Use.

- LBMEpDLL int `lbm_msg_properties_iter_delete` (`lbm_msg_properties_iter_t *iter`)

Deletes an `lbm_msg_properties_iterator`.

- LBMEpDLL int `lbm_msg_properties_iter_first` (`lbm_msg_properties_iter_t *iter`, `lbm_msg_properties_t *properties`)

Begin iterating over an.

- LBMEpDLL int `lbm_msg_properties_iter_next` (`lbm_msg_properties_iter_t *iter`)

Iterate to the next property in an.

- LBMEpDLL int `lbm_src_get_inflight` (`lbm_src_t *src`, `int type`, `int *inflight`, `lbm_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 lbm\_src\_t \*src.*

- LBMEpDLL 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)

*Retrieves the current number of inflight information of a given type from the src pointed to by lbm\_src\_t \*src.*

- LBMEpDLL 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)

*Retrieves the current number of inflight UMQ messages from the ctx pointed to by lbm\_context\_t \*ctx.*

- LBMEpDLL int [lbm\\_ume\\_src\\_msg\\_stable](#) (lbm\_src\_t \*src, lbm\_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.*

- LBMEpDLL int [lbm\\_umq\\_ctx\\_msg\\_stable](#) (lbm\_context\_t \*ctx, const char \*qname, [lbm\\_umq\\_msgid\\_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.*

- LBMEpDLL [lbm\\_ume\\_rcv\\_ack\\_t](#) \* [lbm\\_msg\\_extract\\_ume\\_ack](#) ([lbm\\_msg\\_t](#) \*msg)

*Retrieves the ack structure from a UMP message.*

- LBMEpDLL int [lbm\\_ume\\_ack\\_delete](#) ([lbm\\_ume\\_rcv\\_ack\\_t](#) \*ack)

*Deletes an ack structure.*

- LBMEpDLL int [lbm\\_ume\\_ack\\_send\\_explicit\\_ack](#) ([lbm\\_ume\\_rcv\\_ack\\_t](#) \*ack, lbm\_uint\_t sqn)

*Sends an explicit ack up to the sequence number provided.*

- LBMEpDLL int [lbm\\_ctx\\_umq\\_queue\\_topic\\_list](#) (lbm\_context\_t \*ctx, const char \*queue\_name, [lbm\\_async\\_operation\\_func\\_t](#) \*async\_opfunc)

*Retrieves a list of currently available topics from a queue (asynchronous operation).*

- LBMEpDLL int [lbm\\_umq\\_msg\\_selector\\_create](#) ([lbm\\_umq\\_msg\\_selector\\_t](#) \*\*selector, char \*str, lbm\_uint16\_t len)

*Create an umq message selector (an `lbm_umq_msg_selector_t` structure).*

- LBMEExpDLL int `lbm_umq_msg_selector_delete` (`lbm_umq_msg_selector_t *selector`)  
*Delete the `lbm_umq_msg_selector_t` object previously created with `lbm_umq_msg_selector_create`.*
- LBMEExpDLL int `lbm_rcv_umq_queue_msg_list` (`lbm_rcv_t *rcv`, const char `*queue_name`, `lbm_umq_msg_selector_t *selector`, `lbm_async_operation_func_t *async_opfunc`)  
*Retrieves a list of all currently-queued messages from a queue (asynchronous operation).*
- LBMEExpDLL int `lbm_rcv_umq_queue_msg_retrieve` (`lbm_rcv_t *rcv`, const char `*queue_name`, `lbm_umq_msgid_t *msgids`, int `num_msgids`, `lbm_async_operation_func_t *async_opfunc`)  
*Retrieves a set of queued messages from the queue (asynchronous operation).*
- LBMEExpDLL int `lbm_async_operation_status` (`lbm_async_operation_handle_t handle`, int `flags`)  
*Query the current status of an outstanding asynchronous operation.*
- LBMEExpDLL int `lbm_async_operation_cancel` (`lbm_async_operation_handle_t handle`, int `flags`)  
*Cancel an outstanding asynchronous operation.*
- 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`)  
*Set the user's credential and authentication requirement.*
- LBMEExpDLL int `lbm_authstorage_open_storage_xml` (char `*filename`)  
*Create the storage object from XML password file.*
- LBMEExpDLL void `lbm_authstorage_close_storage_xml` (void)  
*Release the storage object.*
- LBMEExpDLL int `lbm_authstorage_checkpermission` (char `*username`, char `*command`)  
*Check if the user is authorized to execute the specified command.*
- LBMEExpDLL int `lbm_authstorage_addtpnam` (const char `*username`, const char `*pass`, unsigned char `flags`)  
*Add the new user credential to the password file.*

- LBMEpDLL int [lbm\\_authstorage\\_deltplnam](#) (const char \*username)  
*Delete the user credential from the password file.*
- LBMEpDLL int [lbm\\_authstorage\\_user\\_add\\_role](#) (const char \*username, const char \*role)  
*Add one role entry for the user to the password file.*
- LBMEpDLL int [lbm\\_authstorage\\_user\\_del\\_role](#) (const char \*username, const char \*role)  
*Delete the role entry for the user from the password file.*
- LBMEpDLL int [lbm\\_authstorage\\_load\\_roletable](#) ()  
*Load the role table from the password file.*
- LBMEpDLL int [lbm\\_authstorage\\_unload\\_roletable](#) ()  
*Unload the role table.*
- LBMEpDLL int [lbm\\_authstorage\\_roletable\\_add\\_role\\_action](#) (const char \*rolename, const char \*action)  
*Add a new authorized action for the specified role.*
- LBMEpDLL int [lbm\\_authstorage\\_print\\_roletable](#) ()  
*Print the role table saved in the internal data object.*
- LBMEpDLL int [lbm\\_set\\_umm\\_info](#) (lbm\_umm\_info\_t \*info)  
*Connect to and retrieve configuration from a UMM daemon.*
- LBMEpDLL int [lbm\\_is\\_ume\\_capable](#) (void)  
*Determine if the LBM library is capable of UME operations.*
- LBMEpDLL int [lbm\\_is\\_umq\\_capable](#) (void)  
*Determine if the LBM library is capable of UMQ operations.*
- LBMEpDLL void [lbm\\_seterr](#) (int eno, const char \*str)
- LBMEpDLL void [lbm\\_seterrf](#) (int eno, const char \*format,...)
- LBMEpDLL const char \* [lbm\\_strerror](#) (void)
- LBMEpDLL const char \* [lbm\\_strerror\\_errnum](#) (int errnum)
- LBMEpDLL void [lbm\\_sock\\_init](#) ()
- LBMEpDLL lbm\_uint64\_t [lbm\\_create\\_random\\_id](#) ()  
*Create a random id to be used in conjunction with [lbm\\_get\\_jms\\_msg\\_id](#) for JMS compatibility.*

- LBMEExpDLL char \* [lbm\\_get\\_jms\\_msg\\_id](#) (lbm\_uint64\_t source\_id, lbm\_uint64\_t seqno\_id, char \*topic)  
*Create JMS message ID.*
- LBMEExpDLL int [lbm\\_src\\_buff\\_acquire](#) (lbm\_src\_t \*const src, void \*\*const bufp, const size\_t len, const int flags)  
*Acquires a pointer to a buffer of the specified length, to be filled in and sent later.*
- LBMEExpDLL int [lbm\\_src\\_buffs\\_complete](#) (lbm\_src\_t \*const src)  
*Sends all buffers on a transport session that had been previously been acquired.*
- LBMEExpDLL int [lbm\\_src\\_buffs\\_complete\\_and\\_acquire](#) (lbm\_src\_t \*const src, void \*\*const bufp, const size\_t len, const int flags)  
*First sends all buffers on a transport session that had been previously acquired, and then acquires a pointer to a buffer of the specified length, to be filled in and sent later. Equivalent to calling [lbm\\_src\\_buffs\\_complete](#) followed by [lbm\\_src\\_buff\\_acquire](#); included for convenience.*
- LBMEExpDLL int [lbm\\_src\\_buffs\\_cancel](#) (lbm\_src\_t \*const src)  
*Cancels all outstanding (not yet completed) buffers previously acquired using [lbm\\_src\\_buff\\_acquire](#) for this source. All such acquired-but-not-completed buffers for this source (only) will no longer be received by any receivers.*

### 8.1.1 Detailed Description

**Author:**

Todd L. Montgomery - Informatica Corporation.

**Version:**

//UMprod/REL\_6\_7\_1/29West/lbm/src/lib/lbm/lbm.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\_MSG\_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 `lbm_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**

[lbm\\_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 UMP 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**

lbm\_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. ([lbm\\_src\\_send\(\)](#), [lbm\\_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**

lbm\_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. ([lbm\\_src\\_send\(\)](#), [lbm\\_src\\_send\\_ex\(\)](#) etc) : Message ends a batch of messages

**8.1.2.70 #define LBM\_MSG\_EOS 1**

lbm\_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**

lbm\_msg\_t flags. For internal use only.

**8.1.2.72 #define LBM\_MSG\_FLAG\_END\_BATCH 0x2**

lbm\_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\_msg\_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**

lbm\_msg\_t flags. Message has no topic.

**8.1.2.84 #define LBM\_MSG\_FLAG\_UME\_RETRANSMIT 0x8**

lbm\_msg\_t flags. Message is a UMP retransmission.

**8.1.2.85 #define LBM\_MSG\_FLAG\_UME\_SRC\_REGID 0x4000**

lbm\_msg\_t flags. For internal use only.

**8.1.2.86 #define LBM\_MSG\_FLAG\_UMQ\_REASSIGNED 0x40**

lbm\_msg\_t flags. Message is a UMQ message that has been re-assigned at least once.

**8.1.2.87 #define LBM\_MSG\_FLAG\_UMQ\_RESUBMITTED 0x80**

lbm\_msg\_t flags. Message is a UMQ message that has been resubmitted at least once.

**8.1.2.88 #define LBM\_MSG\_FLUSH 0x4**

Flag passed to a source send call E.G. (`lbm_src_send()`, `lbm_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.

**8.1.2.89 #define LBM\_MSG\_HF\_RESET 27**

lbm\_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.

**8.1.2.90 #define LBM\_MSG\_IOV\_GATHER 0x40**

Flag passed to a source send vectored call E.G. ([lbm\\_src\\_sendv\(\)](#), [lbm\\_src\\_sendv\\_ex\(\)](#) etc) : iovec elements should be gather into a single message

**8.1.2.91 #define LBM\_MSG\_NO\_SOURCE\_NOTIFICATION 6**

lbm\_msg\_t type. Notification that no source has been found for topic. Still querying for topic source.

**8.1.2.92 #define LBM\_MSG\_PROPERTIES\_MAX\_NAMELEN 250**

Maximum size for the name of a message property

**8.1.2.93 #define LBM\_MSG\_PROPERTY\_BOOLEAN 0x1**

Message property of boolean type

**8.1.2.94 #define LBM\_MSG\_PROPERTY\_BYTE 0x2**

Message property of byte type

**8.1.2.95 #define LBM\_MSG\_PROPERTY\_DOUBLE 0x7**

Message property of double type

**8.1.2.96 #define LBM\_MSG\_PROPERTY\_FLOAT 0x6**

Message property of float type

**8.1.2.97 #define LBM\_MSG\_PROPERTY\_INT 0x4**

Message property of int type (4 bytes)

**8.1.2.98 #define LBM\_MSG\_PROPERTY\_LONG 0x5**

Message property of long type (8 bytes)

**8.1.2.99 #define LBM\_MSG\_PROPERTY\_NONE 0x0**

Message property with no type (used to indicate an iterator has reached the last element)

**8.1.2.100 #define LBM\_MSG\_PROPERTY\_SHORT 0x3**

Message property of short type (2 bytes)

**8.1.2.101 #define LBM\_MSG\_PROPERTY\_STRING 0x8**

Message property of string type

**8.1.2.102 #define LBM\_MSG\_REQUEST 2**

lbm\_msg\_t type. Request message from source.

**8.1.2.103 #define LBM\_MSG\_RESPONSE 3**

lbm\_msg\_t type. Response message from requestee

**8.1.2.104 #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

**8.1.2.105 #define LBM\_MSG\_UME\_DEREGISTRATION\_COMPLETE\_EX 13**

lbm\_msg\_t type. UMP receiver notification of deregistration complete.

**8.1.2.106 #define LBM\_MSG\_UME\_DEREGISTRATION\_SUCCESS\_EX 12**

lbm\_msg\_t type. UMP receiver notification of deregistration success. Data holds registration IDs, etc.

**8.1.2.107 #define LBM\_MSG\_UME\_DEREGISTRATION\_SUCCESS\_EX\_FLAG\_RPP 0x1**

Deregistration was flagged as coming from a RPP store

**8.1.2.108 #define LBM\_MSG\_UME\_REGISTRATION\_CHANGE 9**

lbm\_msg\_t type. UMP receiver notification of source registration change. Data holds info message.

**8.1.2.109 #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.

**8.1.2.110 #define LBM\_MSG\_UME\_REGISTRATION\_COMPLETE\_EX\_-  
FLAG\_QUORUM 0x1**

Registration completed with only quorum reached.

**8.1.2.111 #define LBM\_MSG\_UME\_REGISTRATION\_COMPLETE\_EX\_-  
FLAG\_RXREQMAX 0x2**

Registration completed with RX REQ maximum used.

**8.1.2.112 #define LBM\_MSG\_UME\_REGISTRATION\_COMPLETE\_EX\_-  
FLAG\_SRC\_SID 0x4**

The src\_session\_id field of the lbm\_msg\_ume\_registration\_complete\_ex\_t structure is valid

**8.1.2.113 #define LBM\_MSG\_UME\_REGISTRATION\_ERROR 7**

lbm\_msg\_t type. UMP receiver registration encountered an error. Data holds error message.

**8.1.2.114 #define LBM\_MSG\_UME\_REGISTRATION\_SUCCESS 8**

lbm\_msg\_t type. UMP receiver registration successful. Data holds registration IDs.

**8.1.2.115 #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.

**8.1.2.116 #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.

**8.1.2.117 #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.

**8.1.2.118 #define LBM\_MSG\_UME\_REGISTRATION\_SUCCESS\_EX\_-  
FLAG\_RPP 0x4**

Registration was flagged as coming from a store that has allowed a RPP receiver

**8.1.2.119 #define LBM\_MSG\_UME\_REGISTRATION\_SUCCESS\_EX\_-  
FLAG\_SRC\_SID 0x8**

The src\_session\_id field of the lbm\_msg\_ume\_registration\_ex\_t structure is valid

**8.1.2.120 #define LBM\_MSG\_UMQ\_DEREGISTRATION\_COMPLETE\_-  
EX 19**

lbm\_msg\_t type. UMQ receiver notification of de-registration completion. Data holds Queue information, etc.

**8.1.2.121 #define LBM\_MSG\_UMQ\_DEREGISTRATION\_COMPLETE\_-  
EX\_FLAG\_ULB 0x1**

Deregistration completed for UMQ ULB source.

**8.1.2.122 #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNED\_EX 24**

lbm\_msg\_t type. UMQ receiver notification of beginning of index.

**8.1.2.123 #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNED\_EX\_FLAG\_-  
REQUESTED 0x2**

Beginning of index assignment that was requested by receiver.

**8.1.2.124 #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNED\_EX\_FLAG\_ -  
ULB 0x1**

Beginning of index from ULB source.

**8.1.2.125 #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.

**8.1.2.126 #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.

**8.1.2.127 #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNMENT\_ -  
ELIGIBILITY\_START\_COMPLETE\_EX\_FLAG\_ULB 0x1**

Index assignment started for ULB source.

**8.1.2.128 #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.

**8.1.2.129 #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNMENT\_ -  
ELIGIBILITY\_STOP\_COMPLETE\_EX\_FLAG\_ULB 0x1**

Index assignment stopped for ULB source.

**8.1.2.130 #define LBM\_MSG\_UMQ\_INDEX\_ASSIGNMENT\_ERROR 26**

lbm\_msg\_t type. UMQ receiver notification of an index assignment error.

**8.1.2.131 #define LBM\_MSG\_UMQ\_INDEX\_RELEASED\_EX 25**

lbm\_msg\_t type. UMQ receiver notification of end of index.

**8.1.2.132 #define LBM\_MSG\_UMQ\_INDEX\_RELEASED\_EX\_FLAG\_ -  
ULB 0x1**

End of index from ULB source.

**8.1.2.133 #define LBM\_MSG\_UMQ\_REASSIGN\_FLAG\_DISCARD 0x1**

Instead of requesting reassignment, request the message be discarded.

**8.1.2.134 #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.

**8.1.2.135 #define LBM\_MSG\_UMQ\_REGISTRATION\_COMPLETE\_EX\_ -  
FLAG\_QUORUM 0x1**

Registration completed with only quorum reached.

**8.1.2.136 #define LBM\_MSG\_UMQ\_REGISTRATION\_COMPLETE\_EX\_ -  
FLAG\_ULB 0x2**

Registration completed for UMQ ULB source.

**8.1.2.137 #define LBM\_MSG\_UMQ\_REGISTRATION\_ERROR 16**

lbm\_msg\_t type. UMQ receiver registration encountered an error. Data holds error message.

**8.1.2.138 #define LBM\_MSG\_UNRECOVERABLE\_LOSS 4**

lbm\_msg\_t type. Missing message detected and not recovered in given time.

**8.1.2.139 #define LBM\_MSG\_UNRECOVERABLE\_LOSS\_BURST 5**

lbm\_msg\_t type. Missing burst of messages detected and not recovered.

**8.1.2.140 #define LBM\_RCV\_BLOCK 0x20**

reserved for future use

**8.1.2.141 #define LBM\_RCV\_BLOCK\_TEMP 0x10**

reserved for future use

**8.1.2.142 #define LBM\_RCV\_NONBLOCK 0x8**

reserved for future use

**8.1.2.143 #define LBM\_RCV\_TOPIC\_STATS\_FLAG\_SRC\_VALID 0x1**

Receiver topic stats structure indicating the source information is valid.

**8.1.2.144 #define LBM\_SRC\_BLOCK 0x20**

Flag passed to a source send call E.G. ([lbm\\_src\\_send\(\)](#), [lbm\\_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.)

**8.1.2.145 #define LBM\_SRC\_BLOCK\_TEMP 0x10**

reserved for future use

**8.1.2.146 #define LBM\_SRC\_COST\_FUNCTION\_REJECT 0xffffffff**

Source cost function return value to indicate this source should be permanently rejected.

**8.1.2.147 #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.

**8.1.2.148 #define LBM\_SRC\_EVENT\_DAEMON\_CONFIRM 4**

Type of source event. For UM daemon usage only, means daemon has confirmed src created

**8.1.2.149 #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.

**8.1.2.150 #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.

**8.1.2.151 #define LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION\_-STATE\_OVER 0x1**

Messages in flight has exceeded the threshold

**8.1.2.152 #define LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION\_-STATE\_UNDER 0x2**

Messages in flight is now below the threshold

**8.1.2.153 #define LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION\_-TYPE\_ULB 0x2**

Specifies a ULB flight size

**8.1.2.154 #define LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION\_-TYPE\_UME 0x1**

Specifies a UMP flight size

**8.1.2.155 #define LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION\_-  
TYPE\_UMQ 0x3**

Specifies a UMQ flight size

**8.1.2.156 #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.

**8.1.2.157 #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.

**8.1.2.158 #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.

**8.1.2.159 #define LBM\_SRC\_EVENT\_UME\_DELIVERY\_-  
CONFIRMATION\_EX\_FLAG\_EXACK 0x8**

Confirmation received with Explicit ACK (EXACK) flagged.

**8.1.2.160 #define LBM\_SRC\_EVENT\_UME\_DELIVERY\_-  
CONFIRMATION\_EX\_FLAG\_OOD 0x4**

Confirmation received with Out-of-Order Delivery (OOD) flagged.

**8.1.2.161 #define LBM\_SRC\_EVENT\_UME\_DELIVERY\_-  
CONFIRMATION\_EX\_FLAG\_UNIQUEACKS 0x1**

Confirmation received for specified number of unique ACKs.

**8.1.2.162 #define LBM\_SRC\_EVENT\_UME\_DELIVERY\_-  
CONFIRMATION\_EX\_FLAG\_UREGID 0x2**

Confirmation received with User Specified Rcv Registration ID flagged.

**8.1.2.163 #define LBM\_SRC\_EVENT\_UME\_DELIVERY\_-  
CONFIRMATION\_EX\_FLAG\_WHOLE\_MESSAGE\_-  
CONFIRMED 0x10**

Whole message (each fragment) has been confirmed

**8.1.2.164 #define LBM\_SRC\_EVENT\_UME\_DEREGISTRATION\_-  
COMPLETE\_EX 32**

Type of source event. For UMP only, means deregistration of source complete (extended form).

**8.1.2.165 #define LBM\_SRC\_EVENT\_UME\_DEREGISTRATION\_-  
SUCCESS\_EX 31**

Type of source event. For UMP only, means deregistration of source successful (extended form).

**8.1.2.166 #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_NOT\_STABLE 33**

Type of source event. For UMP only, means a message is NOT stable, and the source has given up waiting for stability. Event data tells which store the message is not stable at and the reason the source gave up.

**8.1.2.167 #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_NOT\_STABLE\_-  
FLAG\_LOSS 0x40**

Message not stable due to explicitly reported unrecoverable loss at the store or receiver.

**8.1.2.168 #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_NOT\_STABLE\_-  
FLAG\_STORE 0x8**

Message not stable information has store information

**8.1.2.169** `#define LBM_SRC_EVENT_UME_MESSAGE_NOT_STABLE_-  
FLAG_TIMEOUT 0x80`

Message not stable due to stability lifetime timeout.

**8.1.2.170** `#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.

**8.1.2.171** `#define LBM_SRC_EVENT_UME_MESSAGE_RECLAIMED_-  
EX 30`

Type of source event. Message is being reclaimed. Event data holds ACK information.

**8.1.2.172** `#define LBM_SRC_EVENT_UME_MESSAGE_RECLAIMED_EX_-  
FLAG_FORCED 0x1`

Reclaim notification is the result of a forced reclaim

**8.1.2.173** `#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.

**8.1.2.174** `#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.

**8.1.2.175** `#define LBM_SRC_EVENT_UME_MESSAGE_STABLE_EX_-  
FLAG_INTERGROUP_STABLE 0x2`

Message stable for intergroup stability

**8.1.2.176** `#define LBM_SRC_EVENT_UME_MESSAGE_STABLE_EX_-  
FLAG_INTRAGROUP_STABLE 0x1`

Message stable for intragroup stability

**8.1.2.177 #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE\_EX -  
FLAG\_STABLE 0x4**

Message is stable according to behavior desired

**8.1.2.178 #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE\_EX -  
FLAG\_STORE 0x8**

Message stable information has active store information

**8.1.2.179 #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE\_EX -  
FLAG\_USER 0x20**

Message stabilized via lbm\_ume\_src\_msg\_stable API

**8.1.2.180 #define LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE\_EX -  
FLAG\_WHOLE\_MESSAGE\_STABLE 0x10**

Whole message (each fragment) is stable

**8.1.2.181 #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.

**8.1.2.182 #define LBM\_SRC\_EVENT\_UME\_REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_QUORUM 0x1**

Registration completed with only quorum reached.

**8.1.2.183 #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.

**8.1.2.184 #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

**8.1.2.185 #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

**8.1.2.186 #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)

**8.1.2.187 #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.

**8.1.2.188 #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

**8.1.2.189 #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.

**8.1.2.190 #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.

**8.1.2.191 #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.

**8.1.2.192 #define LBM\_SRC\_EVENT\_UMQ\_MESSAGE\_STABLE\_EX -  
FLAG\_INTERGROUP\_STABLE 0x2**

Message stable for intergroup stability

**8.1.2.193 #define LBM\_SRC\_EVENT\_UMQ\_MESSAGE\_STABLE\_EX -  
FLAG\_INTRAGROUP\_STABLE 0x1**

Message stable for intragroup stability

**8.1.2.194 #define LBM\_SRC\_EVENT\_UMQ\_MESSAGE\_STABLE\_EX -  
FLAG\_STABLE 0x4**

Message is stable according to behavior desired

**8.1.2.195 #define LBM\_SRC\_EVENT\_UMQ\_MESSAGE\_STABLE\_EX -  
FLAG\_USER 0x8**

Message stabilized via the lbm\_umq\_ctx\_msg\_stable API

**8.1.2.196 #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.

**8.1.2.197 #define LBM\_SRC\_EVENT\_UMQ\_REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_QUORUM 0x1**

Registration completed with only quorum reached.

**8.1.2.198 #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.

**8.1.2.199 #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.

**8.1.2.200 #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.

**8.1.2.201 #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.

**8.1.2.202 #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.

**8.1.2.203 #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_-  
REASSIGNED\_EX\_FLAG\_EXPLICIT 0x1**

reassignment is the result of the `lbm_msg_umq_reassign` API being called by a receiver

**8.1.2.204 #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-lived. Event data holds message information.

**8.1.2.205 #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_TIMEOUT\_-  
EX\_FLAG\_DISCARD 0x4**

timeout is the result of the `lbm_msg_umq_reassign` API being called with the `LBM_MSG_UMQ_REASSIGN_FLAG_DISCARD` flag set

**8.1.2.206 #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_TIMEOUT\_-  
EX\_FLAG\_EXPLICIT 0x2**

timeout is the result of the `lbm_msg_umq_reassign` API being called by a receiver

**8.1.2.207 #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

**8.1.2.208 #define LBM\_SRC\_EVENT\_UMQ\_ULB\_MESSAGE\_TIMEOUT\_-  
EX\_FLAG\_TOTAL\_LIFETIME\_EXPIRED 0x1**

timeout is the result of the total lifetime expiring

**8.1.2.209 #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.

**8.1.2.210 #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.

**8.1.2.211 #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.

**8.1.2.212 #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-lived. Event data holds receiver information.

**8.1.2.213 #define LBM\_SRC\_EVENT\_WAKEUP 3**

Type of source event. Following earlier return of LBM\_EWOULDBLOCK, means source is ready for more sends

**8.1.2.214 #define LBM\_SRC\_EVENT\_WAKEUP\_FLAG\_MIM 0x2**

Unblocked source is a context-level multicast immediate mode source.

**8.1.2.215 #define LBM\_SRC\_EVENT\_WAKEUP\_FLAG\_NORMAL 0x1**

Unblocked source is a normal (or hot failover) source.

**8.1.2.216 #define LBM\_SRC\_EVENT\_WAKEUP\_FLAG\_REQUEST 0x8**

Unblocked source is a context-level request source.

**8.1.2.217 #define LBM\_SRC\_EVENT\_WAKEUP\_FLAG\_RESPONSE 0x10**

Unblocked source is a context-level response source.

**8.1.2.218 #define LBM\_SRC\_EVENT\_WAKEUP\_FLAG\_UIM 0x4**

Unblocked source is a context-level unicast immediate mode source.

**8.1.2.219 #define LBM\_SRC\_NONBLOCK 0x8**

Flag passed to a source send call E.G. ([lbm\\_src\\_send\(\)](#), [lbm\\_src\\_send\\_ex\(\)](#) etc)  
: If message could not be sent immediately return and error and signal LBM\_EWOULDBLOCK.

**8.1.2.220 #define LBM\_SRC\_SEND\_EX\_FLAG\_APPHDR\_CHAIN 0x8**

Send messages using an appheader chain

**8.1.2.221 #define LBM\_SRC\_SEND\_EX\_FLAG\_CHANNEL 0x20**

Send messages using supplied channel information

**8.1.2.222 #define LBM\_SRC\_SEND\_EX\_FLAG\_HF\_32 0x400**

Send message with the supplied 32 bit hot failover sequence number

**8.1.2.223 #define LBM\_SRC\_SEND\_EX\_FLAG\_HF\_64 0x800**

Send message with the supplied 64 bit hot failover sequence number

**8.1.2.224 #define LBM\_SRC\_SEND\_EX\_FLAG\_HF\_OPTIONAL 0x100**

Send messages marked as an optional message for Hot Failover

**8.1.2.225 #define LBM\_SRC\_SEND\_EX\_FLAG\_PROPERTIES 0x200**

Send message with the supplied messages properties

**8.1.2.226 #define LBM\_SRC\_SEND\_EX\_FLAG\_SEQUENCE\_NUMBER\_-  
INFO 0x2**

Inform application of the sequence numbers used for message

**8.1.2.227 #define LBM\_SRC\_SEND\_EX\_FLAG\_SEQUENCE\_NUMBER\_-  
INFO\_FRAGONLY 0x4**

Inform application of the sequence numbers used for message (fragmented messages only)

**8.1.2.228 #define LBM\_SRC\_SEND\_EX\_FLAG\_UME\_CLIENTD 0x1**

UMP client data pointer is valid

**8.1.2.229 #define LBM\_SRC\_SEND\_EX\_FLAG\_UMQ\_CLIENTD 0x1**

UMQ client data pointer is valid

**8.1.2.230 #define LBM\_SRC\_SEND\_EX\_FLAG\_UMQ\_INDEX 0x40**

Send messages associating them with the supplied UMQ Index

**8.1.2.231 #define LBM\_SRC\_SEND\_EX\_FLAG\_UMQ\_MESSAGE\_ID\_-  
INFO 0x10**

Inform application of the UMQ Message ID used for the message

**8.1.2.232** `#define LBM_SRC_SEND_EX_FLAG_UMQ_TOTAL_-  
LIFETIME 0x80`

umq\_msg\_total\_lifetime is valid

**8.1.2.233** `#define LBM_TOPIC_RES_REQUEST_ADVERTISEMENT 0x04`

Flag passed to [lbm\\_context\\_topic\\_resolution\\_request\(\)](#) to request sources to re-advertise

**8.1.2.234** `#define LBM_TOPIC_RES_REQUEST_CONTEXT_-  
ADVERTISEMENT 0x10`

Flag passed to [lbm\\_context\\_topic\\_resolution\\_request\(\)](#) to request contexts to re-advertise

**8.1.2.235** `#define LBM_TOPIC_RES_REQUEST_CONTEXT_QUERY 0x20`

Flag passed to [lbm\\_context\\_topic\\_resolution\\_request\(\)](#) to request contexts to query for other contexts

**8.1.2.236** `#define LBM_TOPIC_RES_REQUEST_GW_REMOTE_-  
INTEREST 0x40`

Flag passed to [lbm\\_context\\_topic\\_resolution\\_request\(\)](#) to request each Gateway to propagate remote interest.

**8.1.2.237** `#define LBM_TOPIC_RES_REQUEST_QUERY 0x02`

Flag passed to [lbm\\_context\\_topic\\_resolution\\_request\(\)](#) to request receivers to query for source

**8.1.2.238** `#define LBM_TOPIC_RES_REQUEST_RESERVED1 0x08`

reserved for internal use

**8.1.2.239** `#define LBM_TOPIC_RES_REQUEST_WILDCARD_QUERY 0x01`

Flag passed to [lbm\\_context\\_topic\\_resolution\\_request\(\)](#) to request wildcard receivers to query for source

**8.1.2.240** `#define LBM_TRANSPORT_STAT_DAEMON 0xFF`

Transport statistic type. UM Daemon is being used

**8.1.2.241** `#define LBM_TRANSPORT_STAT_LBTIPC LBM_TRANSPORT_-  
TYPE_LBTIPC`

Transport statistic type. LBT-IPC transport

**8.1.2.242** `#define LBM_TRANSPORT_STAT_LBTRDMA LBM_-  
TRANSPORT_TYPE_LBTRDMA`

Transport statistic type. LBT-RDMA transport

**8.1.2.243** `#define LBM_TRANSPORT_STAT_LBTRM LBM_TRANSPORT_-  
TYPE_LBTRM`

Transport statistic type. LBT-RM transport

**8.1.2.244** `#define LBM_TRANSPORT_STAT_LBTRU LBM_TRANSPORT_-  
TYPE_LBTRU`

Transport statistic type. LBT-RU transport

**8.1.2.245** `#define LBM_TRANSPORT_STAT_LBTSMX LBM_-  
TRANSPORT_TYPE_LBTSMX`

Transport statistic type. LBT-SMX transport

**8.1.2.246** `#define LBM_TRANSPORT_STAT_TCP LBM_TRANSPORT_-  
TYPE_TCP`

Transport statistic type. TCP transport

**8.1.2.247** `#define LBM_TRANSPORT_TYPE_LBTIPC 0x40`

Transport type LBT-IPC.

**8.1.2.248 #define LBM\_TRANSPORT\_TYPE\_LBTRDMA 0x20**

Transport type LBT-RDMA.

**8.1.2.249 #define LBM\_TRANSPORT\_TYPE\_LBTRM 0x10**

Transport type LBT-RM.

**8.1.2.250 #define LBM\_TRANSPORT\_TYPE\_LBTRU 0x01**

Transport type LBT-RU.

**8.1.2.251 #define LBM\_TRANSPORT\_TYPE\_LBTSMX 0x4**

Transport type LBT-SMX.

**8.1.2.252 #define LBM\_TRANSPORT\_TYPE\_TCP 0x00**

Transport type TCP.

**8.1.2.253 #define LBM\_UMM\_INFO\_FLAGS\_USE\_SSL 0x1**

Use SSL for UMM daemon connections.

**8.1.2.254 #define LBM\_UMQ\_INDEX\_FLAG\_NUMERIC 0x1**

lbm\_umq\_index\_info\_t flags. Index is a 64-bit unsigned integer.

**8.1.2.255 #define LBM\_WILDCARD\_RCV\_PATTERN\_TYPE\_APP\_CB 3**

Application defined callback pattern type

**8.1.2.256 #define LBM\_WILDCARD\_RCV\_PATTERN\_TYPE\_PCRE 1**

PCRE (Perl Compatible Regular Expressions) pattern type

**8.1.2.257 #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 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 (`lbm_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 (`lbm_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 lbm_context_src_event_func_t_stct` `lbm_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 [lbm\\_context\\_stats\\_t](#) [stct lbm\\_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(\*) [lbm\\_daemon\\_event\\_cb\\_proc](#)([lbm\\_context\\_t](#) \*ctx, int event, const char \*info, void \*clientd)

Set by [lbm\\_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 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 [lbm\\_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 [lbm\\_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 [lbm\\*\\_flight\\_size\\_set\\_inflight\(\)](#).

#### Parameters:

*inflight* Gives the current inflight value.

*clientd* Client data pointer supplied in the call to `lbm_*_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 `lbm_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 `lbm_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 `lbm_context_rcv_immediate_msgs()`.

**Returns:**

0 always.

**8.1.3.18** `typedef struct lbm_iovec_t_stct lbm_iovec_t`

UM replacement for struct iovec for portability.

**8.1.3.19** `typedef struct lbm\_ipv4\_address\_mask\_t\_stct  
lbm\_ipv4\_address\_mask\_t`

A structure used with options to set/get specific addresses within a range.

**8.1.3.20** `typedef int(*) lbm\_log\_cb\_proc(int level, const char *message, void *clientd)`

Set by [lbm\\_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 [lbm\\_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:**

[lbm\\_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 [lbm\\_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 [lbm\\_msg\\_retrieve\\_fragment\\_info\(\)](#).

**8.1.3.25** `typedef struct lbm\_msg\_gateway\_info\_t\_stct lbm\_msg\_gateway\_info\_t`**Deprecated****8.1.3.26** `typedef struct lbm\_msg\_ume\_deregistration\_ex\_t\_stct  
lbm\_msg\_ume\_deregistration\_ex\_t`

A structure used with UM receivers to indicate successful deregistration (extended form).

**8.1.3.27** `typedef struct lbm\_msg\_ume\_registration\_complete\_ex\_t\_stct  
lbm\_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 lbm\_msg\_ume\_registration\_ex\_t\_stct  
lbm\_msg\_ume\_registration\_ex\_t`

A structure used with UM receivers to indicate successful registration (extended form).

**8.1.3.29** `typedef struct lbm\_msg\_ume\_registration\_t\_stct  
lbm\_msg\_ume\_registration\_t`

A structure used with UMP receivers to indicate successful registration.

**8.1.3.30** `typedef struct lbm\_msg\_umq\_deregistration\_complete\_ex\_t\_stct  
lbm\_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 lbm\_msg\_umq\_index\_assigned\_ex\_t\_stct  
lbm\_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 [lbm\\_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 [lbm\\_msg\\_retain\(\)](#) from the receive callback before it returns. It then becomes the application's responsibility to delete the message object using [lbm\\_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 [lbm\\_rcv\\_create\(\)](#).

**Returns:**

0 always.

**8.1.3.37** `typedef void*(*) lbm\_rcv\_src\_notification\_create\_function\_cb(const char *source_name, void *clientd)`

Set by [lbm\\_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:**

[lbm\\_rcv\\_src\\_notification\\_func\\_t](#)

**Parameters:**

*source\_name* Name of the source

*clientd* Client data pointer supplied in the `lbm_rcv_src_notification_func_t` passed to `lbm_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(*) lbm_rcv_src_notification_delete_function_cb(const char *source_name, void *clientd, void *source_clientd)`

Set by `lbm_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:**

[lbm\\_rcv\\_src\\_notification\\_func\\_t](#)

**Parameters:**

*source\_name* Name of the source

*clientd* Client data pointer supplied in the `lbm_rcv_src_notification_func_t` passed to `lbm_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_topic_stats_t_stct lbm_rcv_topic_stats_t`

THIS STRUCTURE IS UNSUPPORTED.

#### 8.1.3.41 `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.

#### 8.1.3.42 `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.

#### 8.1.3.43 `typedef int(*) lbm_request_cb_proc(lbm_request_t *req, lbm_msg_t *msg, void *clientd)`

Set by `lbm_send_request()`, `lbm_multicast_immediate_request()`, `lbm_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 `lbm_send_request()`, etc.

#### Returns:

0 always.

#### 8.1.3.44 `typedef int(*) lbm_src_cb_proc(lbm_src_t *src, int event, void *ed, void *clientd)`

Set by `lbm_src_create()` and `lbm_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 LBM\_SRC\_EVENT\_\* indicating event type.

*ed* Pointer to event data, content dependent on event type.

- For *event* == LBM\_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 [lbm\\_transport\\_source\\_info\\_t\\_stct](#).
- For *event* == LBM\_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* == LBM\_SRC\_EVENT\_WAKEUP, *ed* should be re-cast as a (lbm\_src\_event\_wakeup\_t) and indicates which source has become unblocked.
- For *event* == LBM\_SRC\_EVENT\_SEQUENCE\_NUMBER\_INFO, *ed* should be re-cast as a (lbm\_src\_event\_sequence\_number\_info\_t \*) to extract the sequence number information.
- For *event* == LBM\_SRC\_EVENT\_UME\_REGISTRATION\_ERROR, *ed* should be re-cast as a (const char \*) to extract the error message.
- For *event* == LBM\_SRC\_EVENT\_UME\_REGISTRATION\_SUCCESS, *ed* should be re-cast as a (lbm\_src\_event\_ume\_registration\_t \*) to extract the registration information.
- For *event* == LBM\_SRC\_EVENT\_UME\_REGISTRATION\_SUCCESS\_EX, *ed* should be re-cast as a (lbm\_src\_event\_ume\_registration\_ex\_t \*) to extract the extra registration information.
- For *event* == LBM\_SRC\_EVENT\_UME\_REGISTRATION\_COMPLETE\_EX, *ed* should be re-cast as a (lbm\_src\_event\_ume\_registration\_complete\_ex\_t \*) to extract the extra registration completion information.
- For *event* == LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE, *ed* should be re-cast as a (lbm\_src\_event\_ume\_ack\_info\_t \*) to extract the UMP message acknowledgment information.
- For *event* == LBM\_SRC\_EVENT\_UME\_MESSAGE\_STABLE\_EX, *ed* should be re-cast as a (lbm\_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 (lbm\_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 (lbm\_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 (`lbm_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 (`lbm_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 (`lbm_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 `lbm_src_create()`, etc.

**Returns:**

0 always

**8.1.3.45** `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)`

Set via the "source\_cost\_evaluation\_function" context attribute.

**Parameters:**

*topic* Topic for which the new source was discovered.

*transport* Pointer to a `lbm_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`.

**8.1.3.46** `typedef struct lbm_src_event_flight_size_notification_t_stct lbm_src_event_flight_size_notification_t`

A structure used to indicate a state change in flight size status

**8.1.3.47** `typedef struct lbm\_src\_event\_sequence\_number\_info\_t\_stct  
lbm\_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:**

[lbm\\_src\\_send\\_ex](#)

**8.1.3.48** `typedef struct lbm\_src\_event\_ume\_ack\_ex\_info\_t\_stct  
lbm\_src\_event\_ume\_ack\_ex\_info\_t`

A structure used with UMP sources to indicate message acknowledgment by the store and UMP receivers or message stability information (extended form).

**8.1.3.49** `typedef struct lbm\_src\_event\_ume\_ack\_info\_t\_stct  
lbm\_src\_event\_ume\_ack\_info\_t`

A structure used with UMP sources to indicate message acknowledgment by the store and UMP receivers.

**8.1.3.50** `typedef struct lbm\_src\_event\_ume\_deregistration\_ex\_t\_stct  
lbm\_src\_event\_ume\_deregistration\_ex\_t`

A structure used with UMP sources to indicate successful deregistration (extended form).

**8.1.3.51** `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.

**8.1.3.52** `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).

**8.1.3.53** `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.

**8.1.3.54** `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.

**8.1.3.55** `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.

**8.1.3.56** `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.

**8.1.3.57** `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.

**8.1.3.58** `typedef struct lbm\_src\_event\_umq\_ulb\_receiver\_info\_ex\_t\_stct  
lbm\_src\_event\_umq\_ulb\_receiver\_info\_ex\_t`

A structure used with UMQ ULB source applications to indicate receiver events.

**8.1.3.59** `typedef struct lbm\_src\_event\_wakeup\_t\_stct lbm\_src\_event\_wakeup\_t`

A structure used to indicate the type of source that is now unblocked.

**8.1.3.60** `typedef struct lbm\_src\_notify\_func\_t\_stct lbm\_src\_notify\_func\_t`

A structure used with options to set/get a specific callback information

**8.1.3.61** `typedef int(*) lbm\_src\_notify\_function\_cb(const char *topic\_str, const char *src\_str, void *clientd)`

Set by `lbm\_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:

[lbm\\_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 [lbm\\_transport\\_source\\_info\\_t\\_stct](#).

*clientd* Client data pointer supplied in the `lbm\_src\_notify\_func\_t` passed to the `lbm\_context\_attr\_setopt\(\)`.

Returns:

0 always.

**8.1.3.62** `typedef struct lbm\_src\_send\_ex\_info\_t\_stct lbm\_src\_send\_ex\_info\_t`

See also:

[lbm\\_src\\_send\\_ex](#)

**8.1.3.63** `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.

**8.1.3.64** `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.

**8.1.3.65** 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.

**8.1.3.66** typedef [lbm\\_ulong\\_t](#)(\*) [lbm\\_str\\_hash\\_function\\_cb](#)(const char \*str)

Set by [lbm\\_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

**8.1.3.67** typedef [lbm\\_ulong\\_t](#)(\*) [lbm\\_str\\_hash\\_function\\_cb\\_ex](#)(const char \*str, size\_t strlen, void \*clientd)

Set by [lbm\\_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 [lbm\\_context\\_attr\\_setopt\(\)](#).

**Returns:**

hash value 0..([lbm\\_ulong\\_t](#))-1

**8.1.3.68** typedef int(\*) [lbm\\_timer\\_cb\\_proc](#)([lbm\\_context\\_t](#) \*ctx, const void \*clientd)

Set by [lbm\\_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 `lbm_schedule_timer()`.

**Returns:**

0 always.

**8.1.3.69 typedef struct [lbm\\_timeval\\_t\\_stct](#) [lbm\\_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.

**8.1.3.70 typedef struct [lbm\\_transport\\_source\\_info\\_t\\_stct](#) [lbm\\_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:session\_id[topic\_idx] (session\_id optional, per configuration option `transport_tcp_use_session_id`)  
example: TCP:192.168.0.4:45789:f1789bcc[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](#)

### 8.1.3.71 `typedef struct lbm_ucast_resolver_entry_t_stct lbm_ucast_resolver_entry_t`

A structure used with options to get/set information about unicast resolver daemons.

### 8.1.3.72 `typedef void*(*) lbm_ume_ctx_rcv_ctx_notification_create_ function_cb(const ume_liveness_receiving_context_t *rcv, void *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_rcv_ctx_notification_func_t`

#### Parameters:

*const* struct ume\_liveness\_receiving\_context\_t

*clientd* Client data pointer supplied in the `lbm_ume_rcv_ctx_notification_func_t` passed to `lbm_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.

### 8.1.3.73 `typedef int(*) lbm_ume_ctx_rcv_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_rcv_ctx_notification_func_t`

#### Parameters:

*const* struct lbm\_ume\_liveness\_rcv\_context\_t

*clientd* Client data pointer supplied in the `lbm_ume_ctx_rcv_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.

**8.1.3.74** `typedef struct lbm\_ume\_ctx\_rcv\_ctx\_notification\_func\_t\_stct  
lbm\_ume\_ctx\_rcv\_ctx\_notification\_func\_t`

A Structure used with options to set/get a specific callback function

**8.1.3.75** `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](#)

**8.1.3.76** `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

**8.1.3.77** `typedef int(*) lbm\_ume\_rcv\_recovery\_info\_ex\_function\_-  
cb(lbm\_ume\_rcv\_recovery\_info\_ex\_func\_info\_t *info, void  
*clientd)`

Set by [lbm\\_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:**

[lbm\\_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 [lbm\\_rcv\\_topic\\_attr\\_setopt\(\)](#) with the "ume\_recovery\_sequence\_number\_info\_function" attribute.

**Returns:**

0 always

**8.1.3.78** `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:**

[lbm\\_ume\\_rcv\\_regid\\_func\\_t](#)

**8.1.3.79** `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

**8.1.3.80** `typedef lbm\_uint\_t(*) lbm\_ume\_rcv\_regid\_ex\_function\_-  
cb(lbm\_ume\_rcv\_regid\_ex\_func\_info\_t *info, void  
*clientd)`

Set by [lbm\\_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.

**8.1.3.81** `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

**8.1.3.82** `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.

**8.1.3.83** `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

**8.1.3.84** `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.

**8.1.3.85** `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

**8.1.3.86** `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

**8.1.3.87** `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

**8.1.3.88** `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.

**8.1.3.89** `typedef struct lbm\_umq\_msg\_total\_lifetime\_info\_t\_stct  
lbm\_umq\_msg\_total\_lifetime\_info\_t`

A structure used with UMQ sources to specify a message's total lifetime.

**8.1.3.90** `typedef struct lbm\_umq\_msgid\_t\_stct lbm\_umq\_msgid\_t`**See also:**

[lbm\\_umq\\_regid\\_t](#) A structure used with UMQ messages to identify a message uniquely.

**8.1.3.91** `typedef struct lbm\_umq\_queue\_entry\_t\_stct lbm\_umq\_queue\_entry\_t`

A struct used with options to get/set Registration ID information for UMQ queues

**8.1.3.92** `typedef lbm_uint64_t lbm_umq_regid_t`

Registration ID used for UMQ contexts for both sources and receivers

**8.1.3.93** `typedef struct lbm_umq_ulb_application_set_attr_t_stct  
lbm_umq_ulb_application_set_attr_t`

A struct used with options to get/set UMQ ULB application set attributes

**8.1.3.94** `typedef struct lbm_umq_ulb_receiver_type_attr_t_stct  
lbm_umq_ulb_receiver_type_attr_t`

A struct used with options to get/set UMQ ULB receiver type attributes

**8.1.3.95** `typedef struct lbm_umq_ulb_receiver_type_entry_t_stct  
lbm_umq_ulb_receiver_type_entry_t`

A struct used with options to get/set UMQ ULB Receiver Type entries

**8.1.3.96** `typedef struct lbm_wildcard_rcv_compare_func_t_stct  
lbm_wildcard_rcv_compare_func_t`

A structure used with options to set/get a specific application callback pattern type.

**8.1.3.97** `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.

**8.1.3.98** `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.

**8.1.3.99** `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 [lbm\\_wildcard\\_rcv\\_create\\_func\\_t](#) passed to [lbm\\_wildcard\\_rcv\\_attr\\_setopt\(\)](#) with the "receiver\_create\_callback" attribute.

**Returns:**

Always return 0.

**8.1.3.100** `typedef struct lbm\_wildcard\_rcv\_delete\_func\_t\_stct  
lbm\_wildcard\_rcv\_delete\_func\_t`

A structure used with options to set/get a specific wildcard topic receiver deletion callback type.

**8.1.3.101** `typedef int(*) lbm\_wildcard\_rcv\_delete\_function\_cb(const char *topic\_str, void *clientd)`

Set by [lbm\\_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:**

[lbm\\_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 [lbm\\_wildcard\\_rcv\\_delete\\_func\\_t](#) passed to [lbm\\_wildcard\\_rcv\\_attr\\_setopt\(\)](#) with the "receiver\_delete\_callback" attribute.

**Returns:**

Always return 0.

**8.1.3.102** `typedef struct lbm\_wildcard\_rcv\_stats\_t\_stct lbm\_wildcard\_rcv\_stats\_t`

THIS STRUCTURE IS UNSUPPORTED.

**8.1.3.103** `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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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 LBMEExpDLL 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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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 `lbm_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 `LBMEpDLL int lbm_async_operation_status` (`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 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* overwriting flag.

**Returns:**

negative values for Failure or 0 and positive values for Success.

**8.1.4.15 LBMExpDLL int lbm\_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 lbm\_authstorage\_close\_storage\_xml (void)**

Calling this function will release the storage object created by [lbm\\_authstorage\\_open\\_storage\\_xml\(\)](#).

**Parameters:**

*None.*

**Returns:**

None.

**8.1.4.17 LBMExpDLL int lbm\_authstorage\_deltppnam (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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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 LBMEExpDLL 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 LBMEExpDLL 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 to together). Mask of events to cancel.

##### Returns:

0 for Success and -1 for Failure.

**8.1.4.26** **LBMEpDLL** **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** **LBMEpDLL** **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 LBMEExpDLL 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 LBMEExpDLL 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 LBMExpDLL 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 LBMExpDLL 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 LBMExpDLL 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** LBMEpDLL 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** LBMEpDLL 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** LBMEpDLL 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 LBMEExpDLL 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 LBMEExpDLL 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 LBMEExpDLL 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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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 LBMEExpDLL 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 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 for backward compatibility only. Please pass NULL.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.45 LBMEExpDLL 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 LBMEpDLL int lbm\_context\_delete\_ex (lbm\_context\_t \* ctx, lbm\_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 LBMEpDLL int lbm\_context\_dump (lbm\_context\_t \* ctx, int \* size, lbm\_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 LBMEpDLL lbm\_context\_t\* lbm\_context\_from\_rcv (lbm\_rcv\_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 LBMEExpDLL 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 LBMEExpDLL lbm\_context\_t\* lbm\_context\_from\_wildcard\_rcv (lbm\_wildcard\_rcv\_t \* wrcrv)****Parameters:**

*wrcrv* 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 LBMEExpDLL 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 LBMEExpDLL 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 LBMEpDLL 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 LBMEpDLL 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. For -1, which is returned for critical errors, check the resultant `lbm_errnum` and error message.

**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\_rev\_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** **LBMExpDLL** **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.

**8.1.4.58** **LBMExpDLL** **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** **LBMExpDLL** **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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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** **LBMExpDLL** **int** **lbm\_context\_retrieve\_im\_rcv\_transport\_stats**  
(**lbm\_context\_t** \* *ctx*, **int** \* *num*, **int** *size*, **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.

*size* Size in bytes of each entry in *stats*

*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\_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.

**8.1.4.65** **LBMExpDLL** **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 **lbm\_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 LBMEExpDLL 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 `lbm_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 LBMEExpDLL 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 `lbm_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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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.
- `LBM_TOPIC_RES_REQUEST_CONTEXT_ADVERTISEMENT` - Request context advertisements from quiescent contexts.
- `LBM_TOPIC_RES_REQUEST_CONTEXT_QUERY` - Request context queries from quiescent contexts.
- `LBM_TOPIC_RES_REQUEST_GW_REMOTE_INTEREST` - Request remote interest from Gateways.

*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 LBMEpDLL 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_context_process_events()` to immediately return instead of continuing to process events.

**Parameters:**

*ctx* Pointer to the UM context object.

**8.1.4.75 LBMEpDLL lbm\_uint64\_t lbm\_create\_random\_id ()**

**Returns:**

a random 64 bit long.

**8.1.4.76 LBMEpDLL 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.  
*clientd* Optional client data passed into the proc.

**Returns:**

0 for Success, -1 for failure if the proc returns a negative value.

**8.1.4.77** LBMEExpDLL 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.

This function is deprecated.

**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** LBMEExpDLL 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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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 or NULL for failure.

**8.1.4.82 LBMEpDLL const char\* lbm\_errmsg (void)****Returns:**

Pointer to a static char array holding the error message.

**8.1.4.83 LBMLExpDLL int lbm\_errnum (void)****Returns:**

Integer error number.

**8.1.4.84 LBMLExpDLL 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 LBMLExpDLL 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 LBMLExpDLL int lbm\_event\_queue\_attr\_create (lbm\_event\_queue\_attr\_t \*\* attr)**

The attribute object is allocated and filled with the current default values that are used by lbm\_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 `lbm_event_queue_attr_t` object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.87 LBMEpDLL int lbm\_event\_queue\_attr\_create\_default  
(lbm\_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 `lbm_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 `lbm_event_queue_attr_t` object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.88 LBMEpDLL int lbm\_event\_queue\_attr\_create\_from\_xml  
(lbm\_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 `lbm_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 `lbm_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:**

*eatr* 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 [lbm\\_event\\_queue\\_attr\\_create](#) or [lbm\\_event\\_queue\\_attr\\_create\\_default](#), from which *attr* is initialized.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.92 LBMEpDLL int lbm\_event\_queue\_attr\_getopt**  
(*lbm\_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 LBMEpDLL int lbm\_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 LBMEpDLL int lbm\_event\_queue\_attr\_set\_from\_xml**  
(*lbm\_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** **LBMEpDLL 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** **LBMEpDLL 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** **LBMEpDLL 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** `LBMEpDLL 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`.  
*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** `LBMEpDLL 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 LBMExpDLL 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 LBMExpDLL 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 LBMExpDLL 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 LBMExpDLL lbm\_event\_queue\_t\* lbm\_event\_queue\_from\_wildcard\_rcv (lbm\_wildcard\_rcv\_t \* wrcv)****Parameters:**

*wrcv* 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** **LBMEpDLL int lbm\_event\_queue\_getopt (lbm\_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** **LBMEpDLL int lbm\_event\_queue\_reset\_stats (lbm\_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** **LBMEpDLL int lbm\_event\_queue\_retrieve\_stats (lbm\_event\_queue\_t \* evq, lbm\_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** **LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL int lbm\_hf\_rcv\_create (lbm\_hf\_rcv\_t \*\* hfrcvp, lbm\_context\_t \* ctx, lbm\_topic\_t \* topic, lbm\_rcv\_cb\_proc, void \* clientd, lbm\_event\_queue\_t \* evq)**

**Warning:**

It is not safe to call this function from a context thread callback.

**See also:**

[lbm\\_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 `lbm_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 `lbm_hf_rcv_create()` call should be preceded by a call to `lbm_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 LBMEpDLL int lbm\_hf\_rcv\_delete (lbm\_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 `lbm_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** LBMExpDLL 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** LBMExpDLL 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** LBMExpDLL 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** LBMLExpDLL 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** LBMLExpDLL 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** `LBMExpDLL 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 LBMEpDLL int lbm\_hf\_src\_send\_rcv\_reset (lbm\_src\_t \* src, int flags, lbm\_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 `lbm_hf_src_create` and not `lbm_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 `lbm_src_send_ex_info_t` containing the hf sequence number

**Returns:**

-1 for Failure or 0 for Success

**8.1.4.122 LBMLExpDLL int lbm\_hf\_src\_sendv (lbm\_src\_t \* src, const lbm\_iovec\_t \* iov, int num, lbm\_uint\_t sqn, int flags)**

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 messages where N is the length of the `iovec`; `lbm_hf_src_sendv` will gather the elements of the array into one message.

**See also:**

[lbm\\_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** `LBMEXPDLL int lbm_hf_src_sendv_ex (lbm_src_t * src, const lbm_iovec_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:**

[lbm\\_hf\\_src\\_sendv\\_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

*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 `lbm_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 lbm\_hfx\_attr\_create (lbm\_hfx\_attr\_t \*\* attr)**

The attribute object is allocated and filled with the current default values that are used by lbm\_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 lbm\_hfx\_attr\_t object.

**Returns:**

0 for Success and -1 for Failure

**8.1.4.125 LBMEExpDLL int lbm\_hfx\_attr\_create\_default (lbm\_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 lbm\_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 lbm\_hfx\_attr\_t object.

**Returns:**

0 for Success and -1 for Failure

**8.1.4.126 LBMEExpDLL int lbm\_hfx\_attr\_create\_from\_xml (lbm\_hfx\_attr\_t \*\* attr, const char \* topicname)**

The attribute object is allocated and filled with the current default values that are used by lbm\_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 `lbm_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 LBMEpDLL int lbm\_hfx\_attr\_delete (lbm\_hfx\_attr\_t \* attr)**

The attribute object is cleaned up and deleted.

**Parameters:**

*attr* Pointer to a UM hfx attribute object as returned by [lbm\\_hfx\\_attr\\_create](#).

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.128 LBMEpDLL int lbm\_hfx\_attr\_dump (lbm\_hfx\_attr\_t \* attr, int \* size, lbm\_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 LBMEpDLL int lbm\_hfx\_attr\_dup (lbm\_hfx\_attr\_t \*\* attr, const lbm\_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 `lbm_hfx_attr_t` object.

*original* Pointer to a UM hfx attribute object as returned by [lbm\\_hfx\\_attr\\_create](#) or [lbm\\_hfx\\_attr\\_create\\_default](#), from which *attr* is initialized.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.130 LBMEpDLL int lbm\_hfx\_attr\_getopt (lbm\_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 LBMEpDLL int lbm\_hfx\_attr\_option\_size ()****Returns:**

The number of entries that are of type "hfx"

**8.1.4.132 LBMEpDLL int lbm\_hfx\_attr\_set\_from\_xml (lbm\_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** LBMExpDLL 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** LBMExpDLL 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** `LBMEExpDLL int lbm_hfx_create (lbm_hfx_t ** hfxp,  
lbm_hfx_attr_t * cattr, const char * symbol, lbm_rcv_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** LBMEpDLL 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** LBMEpDLL 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** LBMEpDLL int `lbm_hfx_getopt` (`lbm_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** **LBMExpDLL int lbm\_hfx\_rcv\_create (lbm\_hfx\_rcv\_t \*\* *hfrcvp*, lbm\_hfx\_t \* *hfx*, lbm\_context\_t \* *ctx*, lbm\_rcv\_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\_rcv\_t object. Will be filled in by this function to point to the newly created lbm\_hfx\_rcv\_t object.

*hfx* An lbm\_hfx\_t object created by

**See also:**

[lbm\\_hfx\\_create](#)

**Parameters:**

*ctx* The lbm\_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 lbm\_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** LBMEpDLL int `lbm_hfx_rcv_delete_ex` (`lbm_hfx_rcv_t * hfrcv`,  
`lbm_event_queue_cancel_cb_info_t * cbinfo`)

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.

*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.

**8.1.4.144** LBMEpDLL int `lbm_hfx_rcv_topic_dump` (`lbm_hfx_rcv_t * hfxrcv`,  
`int * size`, `lbm_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 LBMEpDLL int lbm\_hfx\_setopt (lbm\_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 `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* 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 LBMEpDLL int lbm\_hfx\_str\_getopt (lbm\_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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL 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 LBMEpDLL int lbm\_license\_ummmn\_valid ()**

For internal use only. This function tests whether or not MnM is licensed for use.

**Returns:**

0 for Success and -1 for Failure

**8.1.4.153 LBMEpDLL int lbm\_license\_vds\_valid ()**

For internal use only. This function tests whether or not VDS is licensed for use.

**Returns:**

0 for Success and -1 for Failure

**8.1.4.154 LBMEpDLL 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

**8.1.4.155 LBMEpDLL 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.

**8.1.4.156** 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.

**8.1.4.157** LBMExpDLL `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.

**8.1.4.158** LBMExpDLL 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.

**8.1.4.159** **LBMLExpDLL 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.

**8.1.4.160** **LBMLExpDLL 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.

**8.1.4.161** **LBMLExpDLL 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.

**8.1.4.162** LBMExpDLL 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 [lbm\\_errnum\(\)](#) and [lbm\\_errstr\(\)](#).

**8.1.4.163** LBMExpDLL int `lbm_msg_properties_iter_create` (`lbm_msg_properties_iter_t ** iterp`)

**See also:**

[lbm\\_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.

**8.1.4.164 LBMLExpDLL int lbm\_msg\_properties\_iter\_delete  
([lbm\\_msg\\_properties\\_iter\\_t](#) \* *iter*)****See also:**

[lbm\\_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.

**8.1.4.165 LBMLExpDLL 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 [lbm\\_msg\\_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 [lbm\\_msg\\_properties\\_iter\\_first](#) is called to associate it with each new [properties](#) object.

[lbm\\_msg\\_properties\\_iter\\_next](#)

**Parameters:**

*iter* An iterator object allocated via

**See also:**

[lbm\\_msg\\_properties\\_iter\\_create](#)

**Parameters:**

*properties* A [properties](#) object, either retrieved from an [lbm\\_msg\\_t](#), or created via

**See also:**

[lbm\\_msg\\_properties\\_create](#)

**Returns:**

LBM\_OK for success, LBM\_FAILURE if there are no elements contained in the properties object.

**8.1.4.166 LBMEpDLL int lbm\_msg\_properties\_iter\_next  
(lbm\_msg\_properties\_iter\_t \* iter)****See also:**

lbm\_msg\_properties\_t object.  
[lbm\\_msg\\_properties\\_iter\\_first](#)  
[lbm\\_msg\\_properties\\_iter\\_prev](#)

**Parameters:**

*iter* Iterate to the next element in the currently associated lbm\_msg\_properties\_t object.

**Returns:**

LBM\_OK for success, LBM\_FAILURE if the iterator already points to the last element.

**8.1.4.167 LBMEpDLL int lbm\_msg\_properties\_set (lbm\_msg\_properties\_t \*  
properties, const char \* name, const 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 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 [lbm\\_errnum\(\)](#) and [lbm\\_errstr\(\)](#).

**8.1.4.168 LBMEpDLL int lbm\_msg\_retain ([lbm\\_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 [lbm\\_msg\\_delete\(\)](#).

**Parameters:**

*msg* Pointer to a UM message object to retain.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.169 LBMEpDLL 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.

**8.1.4.170 LBMEpDLL int lbm\_msg\_retrieve\_gateway\_info ([lbm\\_msg\\_t](#) \* *msg*, [lbm\\_msg\\_gateway\\_info\\_t](#) \* *info*)**

This function is deprecated.

**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.

**8.1.4.171** LBMEpDLL int lbm\_msg\_retrieve\_msgid ([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.

**8.1.4.172** LBMEpDLL 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.

**8.1.4.173** LBMEpDLL 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.

**8.1.4.174** LBMEpDLL 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.

**8.1.4.175 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.

**8.1.4.176 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.

**8.1.4.177** 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 `lbm_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*.

*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 `lbm_multicast_immediate_message` for more information.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.178** LBMEExpDLL int `lbm_queue_immediate_message` (`lbm_context_t` \* *ctx*, const char \* *qname*, const char \* *topic*, const char \* *data*, `size_t` *len*, int *flags*, `lbm_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 *lbm\_unicast\_immediate\_message* for more information.

*info* Pointer to *lbm\_src\_send\_ex\_info\_t* options

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.179** LBMEpDLL **int** *lbm\_rcv\_create* (*lbm\_rcv\_t* \*\* *rcvp*, *lbm\_context\_t* \* *ctx*, *lbm\_topic\_t* \* *topic*, *lbm\_rcv\_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:**

*rcvp* A pointer to a pointer to a UM receiver object. Will be filled in by this function to point to the newly created *lbm\_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 *lbm\_rcv\_create()* call should be preceded by a call to *lbm\_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.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.180 LBMEpDLL int lbm\_rcv\_delete (lbm\_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 [lbm\\_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.

**8.1.4.181 LBMEpDLL int lbm\_rcv\_delete\_ex (lbm\_rcv\_t \* rcv, lbm\_event\_queue\_cancel\_cb\_info\_t \* cbinfo)**

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.

*cbinfo* Cancellation callback information, containing the event queue, function pointer, and client data for the callback.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.182 LBMEpDLL lbm\_rcv\_t\* lbm\_rcv\_from\_hf\_rcv (lbm\_hf\_rcv\_t \* hfrcv)****Parameters:**

*hfrcv* Pointer to a UM HF receiver object.

**Returns:**

Pointer to a UM receiver for the LBM HF receiver object.

**8.1.4.183 LBMEExpDLL lbm\_rcv\_t\* lbm\_rcv\_from\_hfx\_rcv (lbm\_hfx\_rcv\_t \* hfxrcv)****Parameters:**

*hfxrcv* A pointer to a UM hfx\_rcv\_t object.

**8.1.4.184 LBMEExpDLL int lbm\_rcv\_getopt (lbm\_rcv\_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.

**8.1.4.185 LBMEExpDLL int lbm\_rcv\_msg\_source\_clientd (lbm\_rcv\_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.

**8.1.4.186** `LBMEpDLL int lbm_rcv_reset_all_transport_stats (lbm_rcv_t * rcv)`

**Parameters:**

*rcv* Pointer to the UM receiver to reset statistics for.

**Returns:**

-1 for Failure and 0 for Success.

**8.1.4.187** `LBMEpDLL 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.

**8.1.4.188** `LBMEpDLL 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_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.

**8.1.4.189** LBMEpDLL 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, excluding the topic index, to retrieve stats for. See also option source\_includes\_topic\_index.

*stats* Pointer to a stats structure to fill in.

**Returns:**

-1 for Failure and 0 for Success.

**8.1.4.190** LBMEpDLL 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.

**8.1.4.191** LBMEpDLL 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.

**8.1.4.192 LBMEpDLL 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.

**8.1.4.193 LBMEpDLL 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

**8.1.4.194 LBMEpDLL 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.

**8.1.4.195 LBMEpDLL 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.

**8.1.4.196 LBMEpDLL 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.

**8.1.4.197 LBMEpDLL 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.

**8.1.4.198 LBMEpDLL 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

**8.1.4.199 LBMEpDLL 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.

**8.1.4.200 LBMEpDLL 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.

**8.1.4.201 LBMEpDLL int lbm\_rcv\_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"

**8.1.4.202 LBMEpDLL int lbm\_rcv\_topic\_attr\_set\_from\_xml  
(lbm\_rcv\_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.

**8.1.4.203 LBMLExpDLL 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.

**8.1.4.204 LBMLExpDLL 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.

**8.1.4.205 LBMEpDLL 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.

**8.1.4.206 LBMEpDLL int lbm\_rcv\_topic\_dump** (*lbm\_rcv\_t \* rcv, int \* size, lbm\_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

**8.1.4.207 LBMEpDLL int lbm\_rcv\_topic\_lookup** (*lbm\_topic\_t \*\* topicp, lbm\_context\_t \* ctx, const char \* symbol, const lbm\_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 `lbm_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.

**8.1.4.208 LBMEpDLL int lbm\_rcv\_ume\_deregister (lbm\_rcv\_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

**8.1.4.209 LBMEpDLL int lbm\_rcv\_umq\_deregister (lbm\_rcv\_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.

**8.1.4.210 LBMEpDLL int lbm\_rcv\_umq\_index\_release (lbm\_rcv\_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.

**8.1.4.211 LBMEpDLL int lbm\_rcv\_umq\_index\_reserve (lbm\_rcv\_t \* rcv, const char \* queue\_name, lbm\_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.

**8.1.4.212 LBMEpDLL int lbm\_rcv\_umq\_index\_start\_assignment (lbm\_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.

**8.1.4.213** LBMEExpDLL int `lbm_rcv_umq_index_stop_assignment` (`lbm_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.

**8.1.4.214** LBMEExpDLL int `lbm_rcv_umq_queue_msg_list` (`lbm_rcv_t *rcv, 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.

This function is deprecated.

**Parameters:**

*rcv* 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.

**8.1.4.215** `LBMExpDLL int lbm_rcv_umq_queue_msg_retrieve (lbm_rcv_t * rcv, 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).

This function is deprecated.

**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.

**8.1.4.216** `LBMExpDLL int lbm_rcv_unsubscribe_channel (lbm_rcv_t * rcv, lbm_uint32_t channel)`

Remove a subscription to a channel previously subscribed to with

**See also:**

[lbm\\_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

**8.1.4.217** LBMEpDLL int `lbm_rcv_unsubscribe_channel_ex` (`lbm_rcv_t *rcv`,  
`lbm_uint32_t channel`, `lbm_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

**8.1.4.218** LBMEpDLL 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.

**8.1.4.219 LBMEpDLL 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 `lbm_request_t` object returned by `lbm_send_request`.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.220 LBMEpDLL int lbm\_request\_delete\_ex (lbm\_request\_t \* req, lbm\_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 `lbm_request_t` object returned by `lbm_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.

**8.1.4.221 LBMEpDLL int lbm\_response\_delete (lbm\_response\_t \* resp)**

When an application receives a *request*, the message object *lbm\_msg\_t* contains a response object *response* which is used by [lbm\\_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:**

[lbm\\_msg\\_t](#), [lbm\\_msg\\_delete](#)

**Parameters:**

*resp* A pointer to an *lbm\_response\_t* object given in a *lbm\_msg\_t* object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.222 LBMEpDLL 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.

**8.1.4.223** **LBMEpDLL 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.

**8.1.4.224** **LBMEpDLL int lbm\_send\_request (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)**

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`.
- 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.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.225** `LBMEpDLL 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*.

*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.

#### 8.1.4.226 LBMEpDLL `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.

#### 8.1.4.227 LBMEpDLL `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.

**8.1.4.228 LBMEpDLL 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.

**8.1.4.229 LBMEpDLL 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).

**8.1.4.230 LBMEpDLL 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).

**8.1.4.231 LBMEpDLL 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).

**8.1.4.232 LBMEpDLL 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).

**8.1.4.233 LBMEpDLL void lbm\_set\_uim\_loss\_rate (int rate)**

Set the UIM loss rate. This is equivalent to setting the environment variable LBM\_UIM\_LOSS\_RATE, but allows the loss rate to be changed under program control.

**Parameters:**

*rate* Loss rate (from 0 to 100).

**8.1.4.234 LBMEpDLL 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.

**8.1.4.235** LBMEpDLL int lbm\_src\_buff\_acquire (lbm\_src\_t \*const src, void \*\*const bufp, const size\_t len, const int flags)

**Warning:**

lbm\_src\_buff\_acquire is NOT thread safe between sources on the same transport session

**Parameters:**

*src* The source object

*bufp* A pointer to a pointer to a buffer; the pointer will be set to non-NULL upon success.

*len* The length of the buffer, in bytes, that is being requested.

*flags* Flags field to control blocking (the default) or non-blocking (if LBM\_SRC\_NONBLOCK is set) behavior.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.236** LBMEpDLL int lbm\_src\_buffs\_cancel (lbm\_src\_t \*const src)

**Warning:**

lbm\_src\_buffs\_cancel is NOT thread safe between sources on the same transport session

**Parameters:**

*src* The source object

**Returns:**

0 for Success and -1 for Failure

**8.1.4.237** LBMEpDLL int lbm\_src\_buffs\_complete (lbm\_src\_t \*const src)

**Warning:**

lbm\_src\_buffs\_complete is NOT thread safe between sources on the same transport session

**Parameters:**

*src* The source object

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.238** **LBMEpDLL int lbm\_src\_buffs\_complete\_and\_acquire (lbm\_src\_t \*const src, void \*\*const bufp, const size\_t len, const int flags)**

**Warning:**

lbm\_src\_buff\_acquire is NOT thread safe between sources on the same transport session

**Parameters:**

*src* The source object

*bufp* A pointer to a pointer to a buffer; the pointer will be set to non-NULL upon success.

*len* The length of the buffer, in bytes, that is being requested.

*flags* Flags field to control blocking (the default) or non-blocking (if LBM\_SRC\_NONBLOCK is set) behavior.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.239** **LBMEpDLL 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

**8.1.4.240 LBMEpDLL 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

**8.1.4.241 LBMEpDLL int lbm\_src\_create (lbm\_src\_t \*\* *srp*, 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:**

*srp* 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.

**8.1.4.242 LBMEpDLL 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.

**8.1.4.243 LBMEpDLL 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.

**8.1.4.244 LBMEpDLL 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.

**8.1.4.245** LBMEpDLL int `lbm_src_get_inflight` (`lbm_src_t * src`, int *type*, int \* *inflight*, `lbm_flight_size_set_inflight_cb_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.

**8.1.4.246** LBMEpDLL 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`, 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.

**8.1.4.247** `LBMLExpDLL 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.

**8.1.4.248** `LBMLExpDLL int lbm_src_reset_transport_stats (lbm_src_t * src)`**Parameters:**

- src* Pointer to the UM source to reset statistics for.

**Returns:**

-1 for Failure and 0 for Success.

**8.1.4.249** `LBMLExpDLL int lbm_src_retrieve_transport_stats (lbm_src_t * src, lbm_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.

**8.1.4.250** `LBMLExpDLL int lbm_src_send (lbm_src_t * src, const char * msg, size_t len, int flags)`**Warning:**

Do not call this function from a context thread callback, as this can 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.

**8.1.4.251** `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 `lbm_src_send_ex_info_t` options

**Returns:**

-1 for Failure or 0 for Success.

**8.1.4.252 LBMExpDLL int lbm\_src\_sendv (lbm\_src\_t \* src, const lbm\_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.

**8.1.4.253 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.

**8.1.4.254 LBMEpDLL 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.

**8.1.4.255 LBMEpDLL 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.

**8.1.4.256 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.

**8.1.4.257 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.

**8.1.4.258 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 `lbm_src_topic_attr_t` object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.259 LBMEpDLL int lbm\_src\_topic\_attr\_create\_default  
(lbm\_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 `lbm_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 `lbm_src_topic_attr_t` object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.260 LBMEpDLL int lbm\_src\_topic\_attr\_create\_from\_xml  
(lbm\_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 `lbm_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 `lbm_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.

**8.1.4.261 LBMEpDLL 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.

**8.1.4.262 LBMEpDLL int lbm\_src\_topic\_attr\_dump (lbm\_src\_topic\_attr\_t \* attr, 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

**8.1.4.263 LBMEpDLL 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.

**8.1.4.264 LBMEpDLL 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.

**8.1.4.265 LBMEpDLL 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"

**8.1.4.266 LBMEpDLL 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.

**8.1.4.267 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.

**8.1.4.268 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.

**8.1.4.269 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.

**8.1.4.270 LBMEpDLL 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

**8.1.4.271 LBMEpDLL 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.

**8.1.4.272 LBMEpDLL 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.

**8.1.4.273** LBMEpDLL 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.

**8.1.4.274** LBMEpDLL 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.

**8.1.4.275** LBMEpDLL int lbm\_ume\_ack\_delete (lbm\_ume\_rcv\_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.

**8.1.4.276** LBMEpDLL 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.

**8.1.4.277** LBMEpDLL 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

**8.1.4.278** LBMEpDLL int lbm\_umq\_ctx\_msg\_stable (lbm\_context\_t \* *ctx*,  
const char \* *qname*, lbm\_umq\_msgid\_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

#### 8.1.4.279 LBMExpDLL 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.

#### 8.1.4.280 LBMExpDLL int lbm\_umq\_msg\_selector\_delete (lbm\_umq\_msg\_selector\_t \* selector)

**Parameters:**

*selector* Pointer to lbm\_umq\_msg\_selector\_t object.

#### 8.1.4.281 LBMExpDLL 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" or "TCP:domain: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.

**8.1.4.282** LBMEpDLL int lbm\_unicast\_immediate\_request (lbm\_request\_t \*\* reqp, lbm\_context\_t \* ctx, const char \* target, const char \* topic, const char \* data, size\_t len, lbm\_request\_cb\_proc proc, void \* clientd, lbm\_event\_queue\_t \* evq, int flags)

**Parameters:**

- reqp* A pointer to a pointer for the lbm\_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" or "TCP:domain: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 *lbm\_unicast\_immediate\_message* for more information.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.283** LBMEpDLL const char\* lbm\_version (void)

**Returns:**

String containing version information for UM.

**8.1.4.284** LBMEpDLL int lbm\_wildcard\_rcv\_attr\_create (lbm\_wildcard\_rcv\_attr\_t \*\* attr)

The attribute object is allocated and filled with the current default values that are used by lbm\_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 `lbm_wildcard_rcv_attr_t` object.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.285 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.

**8.1.4.286 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* They type of pattern. Both *pattern\_type* and *pattern* must match in XML configuration.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.287 LBMEpDLL int lbm\_wildcard\_rcv\_attr\_delete  
(lbm\_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 [lbm\\_wildcard\\_rcv\\_attr\\_create](#).

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.288 LBMEpDLL int lbm\_wildcard\_rcv\_attr\_dump  
(lbm\_wildcard\_rcv\_attr\_t \* wattr, int \* size, lbm\_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

**8.1.4.289 LBMEpDLL int lbm\_wildcard\_rcv\_attr\_dup  
(lbm\_wildcard\_rcv\_attr\_t \*\* attr, const lbm\_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 `lbm_wildcard_rcv_attr_t` object.

*original* Pointer to a UM wildcard receiver attribute object as returned by `lbm_wildcard_rcv_attr_create` or `lbm_wildcard_rcv_attr_create_default`, from which *attr* is initialized.

**Returns:**

0 for Success and -1 for Failure.

**8.1.4.290** `LBMExpDLL 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.

**8.1.4.291** `LBMExpDLL 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"

**8.1.4.292** `LBMExpDLL 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 con-

text 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.

**8.1.4.293** `LBMEpDLL int lbm_wildcard_rcv_attr_setopt  
(lbm_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.

**8.1.4.294** `LBMEpDLL int lbm_wildcard_rcv_attr_str_getopt  
(lbm_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.

**8.1.4.295** LBMEpDLL int `lbm_wildcard_rcv_attr_str_setopt`  
(`lbm_wildcard_rcv_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.

**8.1.4.296** LBMEpDLL int `lbm_wildcard_rcv_create` (`lbm_wildcard_rcv_t ** wrcvp`, `lbm_context_t * ctx`, `const char * pattern`, `const lbm_rcv_topic_attr_t * tattr`, `const lbm_wildcard_rcv_attr_t * wattr`, `lbm_rcv_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_rcv_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.

#### 8.1.4.297 LBMLExpDLL int lbm\_wildcard\_rcv\_delete (lbm\_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 [lbm\\_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.

#### 8.1.4.298 LBMLExpDLL int lbm\_wildcard\_rcv\_delete\_ex (lbm\_wildcard\_rcv\_t \* wrcv, lbm\_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.

**8.1.4.299 LBMExpDLL int lbm\_wildcard\_rcv\_dump (lbm\_wildcard\_rcv\_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

**8.1.4.300 LBMExpDLL int lbm\_wildcard\_rcv\_getopt (lbm\_wildcard\_rcv\_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.

**8.1.4.301 LBMEpDLL int lbm\_wildcard\_rcv\_setopt (lbm\_wildcard\_rcv\_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_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* 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.

**8.1.4.302 LBMEpDLL int lbm\_wildcard\_rcv\_str\_getopt (lbm\_wildcard\_rcv\_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.

**8.1.4.303 LBMEpDLL int lbm\_wildcard\_rcv\_str\_setopt (lbm\_wildcard\_rcv\_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 `lbm_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.

**8.1.4.304 LBMEpDLL int lbm\_wildcard\_rcv\_subscribe\_channel**  
(*lbm\_wildcard\_rcv\_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

**8.1.4.305 LBMEpDLL 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.

**8.1.4.306** **LBMEpDLL** **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.

**8.1.4.307** **LBMEpDLL** **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.

**8.1.4.308** **LBMEpDLL** **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.

**8.1.4.309 LBMExpDLL 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

**8.1.4.310 LBMExpDLL int lbm\_wildcard\_rcv\_unsubscribe\_channel\_ex**  
(lbm\_wildcard\_rcv\_t \* *wrcv*, lbm\_uint32\_t *channel*,  
[lbm\\_event\\_queue\\_cancel\\_cb\\_info\\_t](#) \* *cbinfo*)**Parameters:**

*rcv* A pointer to a UM wildcard 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

**8.1.4.311 LBMExpDLL int lbm\_win32\_static\_thread\_attach (void)**

For internal use only. 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

**8.1.4.312 LBMEpDLL int lbm\_win32\_static\_thread\_detach (void)**

For internal use only. 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

**8.1.4.313 LBMEpDLL 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

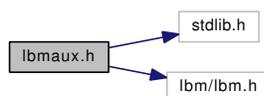
## 8.2 lbmaux.h File Reference

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](#) (lbm\_context\_t \*\*Context, const char \*ConfigFile)  
*Create and initialize an lbm\_context\_t object, initialized with configuration options from a file.*
- LBMEExpDLL int [lbmaux\\_context\\_attr\\_setopt\\_from\\_file](#) (lbm\_context\_attr\_t \*Attributes, const char \*ConfigFile)  
*Set attributes values in an lbm\_context\_attr\_t object from a configuration file.*
- LBMEExpDLL int [lbmaux\\_src\\_topic\\_attr\\_setopt\\_from\\_file](#) (lbm\_src\_topic\_attr\_t \*Attributes, const char \*ConfigFile)  
*Set attributes values in an lbm\_src\_topic\_attr\_t object from a configuration file.*
- LBMEExpDLL int [lbmaux\\_rcv\\_topic\\_attr\\_setopt\\_from\\_file](#) (lbm\_rcv\_topic\_attr\_t \*Attributes, const char \*ConfigFile)  
*Set attributes values in an lbm\_rcv\_topic\_attr\_t object from a configuration file.*
- LBMEExpDLL int [lbmaux\\_wildcard\\_rcv\\_attr\\_setopt\\_from\\_file](#) (lbm\_wildcard\_rcv\_attr\_t \*Attributes, const char \*ConfigFile)  
*Set attributes values in an lbm\_wildcard\_rcv\_attr\_t object from a configuration file.*
- LBMEExpDLL int [lbmaux\\_event\\_queue\\_attr\\_setopt\\_from\\_file](#) (lbm\_event\_queue\_attr\_t \*Attributes, const char \*ConfigFile)  
*Set attributes values in an lbm\_event\_queue\_attr\_t object from a configuration file.*

## 8.2.1 Detailed Description

**Author:**

David K. Ameiss - Informatica Corporation

**Version:**

//UMprod/REL\_6\_7\_1/29West/lbm/src/auxx/lbm/lbmaux.h#1

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 LBMLExpDLL int lbmaux\_context\_create\_from\_file (lbm\_context\_t \*\*  
Context, const char \* ConfigFile)**

This function parses a configuration file, and creates an lbm\_context\_attr\_t object with context-scope option values from the configuration file. It then calls [lbm\\_context\\_create\(\)](#) with the attributes object.

**See also:**

[lbm\\_context\\_create\(\)](#)

[lbm\\_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 lbm\_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 LBMLExpDLL int lbmaux\_event\_queue\_attr\_setopt\_from\_file  
(lbm\_event\_queue\_attr\_t \* Attributes, const char \* ConfigFile)**

This function parses a configuration file, and applies event-queue-scope option values to an lbm\_event\_queue\_attr\_t object.

**Parameters:**

**Attributes** A pointer to an initialized lbm\_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\_rcv\_topic\_attr\_setopt\_from\_file  
(lbm\_rcv\_topic\_attr\_t \* *Attributes*, const char \* *ConfigFile*)**

This function parses a configuration file, and applies receiver-scope option values to an lbm\_rcv\_topic\_attr\_t object.

**Parameters:**

*Attributes* A pointer to an initialized lbm\_rcv\_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  
(lbm\_src\_topic\_attr\_t \* *Attributes*, const char \* *ConfigFile*)**

This function parses a configuration file, and applies source-scope option values to an lbm\_src\_topic\_attr\_t object.

**Parameters:**

*Attributes* A pointer to an initialized lbm\_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\_rcv\_attr\_setopt\_from\_file  
(lbm\_wildcard\_rcv\_attr\_t \* *Attributes*, const char \* *ConfigFile*)**

This function parses a configuration file, and applies wildcard-receiver-scope option values to an lbm\_wildcard\_rcv\_attr\_t object.

**Parameters:**

*Attributes* A pointer to an initialized lbm\_wildcard\_rcv\_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 "lbm/lbm.h"
```

Include dependency graph for lbmht.h:



### Data Structures

- struct [lbm\\_delete\\_cb\\_info\\_t\\_stct](#)

*Structure passed to the [lbm\\_hypertopic\\_rcv\\_delete\(\)](#) function so that a deletion callback may be called.*

### Defines

- #define [LBM\\_HT\\_BASE\\_MATCH\\_LEVEL](#) 0
- #define [LBM\\_HT\\_TOKEN\\_GLOBNAME](#) 0
- #define [LBM\\_HT\\_TOKEN\\_GLOBPATH](#) 1
- #define [LBM\\_HT\\_TOKEN\\_NAME](#) 2
- #define [LBM\\_HT\\_INIT\\_BRANCH\\_SZ](#) 16
- #define [LBM\\_HT\\_CBVEC\\_SZ](#) 16
- #define [LBM\\_HT\\_CBVEC\\_FLAG\\_ACTIVE](#) 1
- #define [LBM\\_HT\\_CBVEC\\_FLAG\\_DELETED](#) 2

### Typedefs

- typedef lbm\_hypertopic\_rcv\_stct [lbm\\_hypertopic\\_rcv\\_t](#)  
*HyperTopic receiver object (opaque).*
- typedef int(\*) [lbm\\_hypertopic\\_rcv\\_cb\\_proc](#) (lbm\_hypertopic\_rcv\_t \*hrcv, [lbm\\_msg\\_t](#) \*msg, void \*clientd)  
*Application callback for messages delivered to HyperTopic receivers.*
- typedef void(\*) [lbm\\_delete\\_cb\\_proc](#) (int dispatch\_thrd, void \*clientd)  
*Application callback for [lbm\\_hypertopic\\_rcv\\_delete\(\)](#).*
- typedef [lbm\\_delete\\_cb\\_info\\_t\\_stct](#) [lbm\\_delete\\_cb\\_info\\_t](#)

Structure passed to the `lbm_hypertopic_rcv_delete()` function so that a deletion callback may be called.

## Functions

- LBMEpDLL int `lbm_hypertopic_rcv_init` (`lbm_hypertopic_rcv_t` \*\*hrcvp, `lbm_context_t` \*ctx, const char \*prefix, `lbm_event_queue_t` \*evq)

*Initialize a HyperTopic receiver.*

- LBMEpDLL int `lbm_hypertopic_rcv_add` (`lbm_hypertopic_rcv_t` \*hrcv, const char \*pattern, `lbm_hypertopic_rcv_cb_proc` proc, void \*clientd)

*Add a topic pattern to the set of topics being received by a HyperTopic receiver.*

- LBMEpDLL 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)

*Delete a previously added topic from a HyperTopic receiver topic set.*

- LBMEpDLL int `lbm_hypertopic_rcv_destroy` (`lbm_hypertopic_rcv_t` \*hrcv)

*Clean-up HyperTopic receiver previously created by `lbm_hypertopic_rcv_init()`.*

### 8.3.1 Detailed Description

#### Author:

M. Garwood - Informatica Corporation

#### Version:

//UMprod/REL\_6\_7\_1/29West/lbm/src/lib/lbm/lbmht.h#1

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 [lbm\\_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 [lbm\\_hypertopic\\_rcv\\_add\(\)](#) and [lbm\\_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 [lbm\\_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 [lbm\\_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(*) lbm_delete_cb_proc(int dispatch_thrd, void *clientd)`

Set by `lbm_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:**

[lbm\\_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 `lbm_hypertopic_rcv_delete()` returns).
- 0 - Called directly by the `lbm_hypertopic_rcv_delete()` function.

*clientd* Client data pointer supplied in the `lbm_delete_cb_info_t` passed to `lbm_hypertopic_rcv_delete()`.

**Returns:**

0 always.

#### 8.3.2.2 `typedef int(*) lbm_hypertopic_rcv_cb_proc(lbm_hypertopic_rcv_t *hrcv, lbm_msg_t *msg, void *clientd)`

Set by `lbm_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 `lbm_msg_retain()` from the receive callback before it returns. It then becomes the application's responsibility to delete the message object by calling `lbm_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 LBMEpDLL 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 LBMEpDLL 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 *pattern*.
- 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** LBMExpDLL 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** LBMExpDLL 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:**

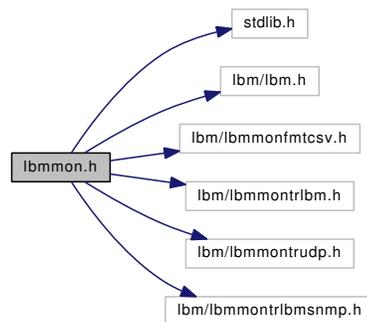
0 for success, -1 on failure

## 8.4 lbmmon.h File Reference

Ultra Messaging (UM) Monitoring API.

```
#include <stdlib.h>
#include <lbm/lbm.h>
#include <lbm/lbmmonfmtcsv.h>
#include <lbm/lbmmontrlbm.h>
#include <lbm/lbmmontrudp.h>
#include <lbm/lbmmontrlbsnmp.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 [lbmmon\\_src\\_statistics\\_func\\_t\\_stct](#)

*A structure that holds the callback information for source statistics.*

- struct [lbmmon\\_evq\\_statistics\\_func\\_t\\_stct](#)

*A structure that holds the callback information for event queue statistics.*

- struct [lbmmon\\_ctx\\_statistics\\_func\\_t\\_stct](#)

*A structure that holds the callback information for context statistics.*

- struct [lbmmon\\_rcv\\_topic\\_statistics\\_func\\_t\\_stct](#)

*For internal use only. A structure that holds the callback information for receiver topic statistics.*

- struct [lbmmon\\_wildcard\\_rcv\\_statistics\\_func\\_t\\_stct](#)

*A structure that holds the callback information for wildcard receiver statistics.*

- struct [lbmmon\\_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\\_EOP](#) (LBMMON\_ERROR\_BASE + 7)
- #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\\_PACKET\\_TYPE\\_RECEIVER\\_TOPIC](#) 4
- #define [LBMMON\\_PACKET\\_TYPE\\_WILDCARD\\_RECEIVER](#) 5
- #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\\_CTXINST](#) 7
- #define [LBMMON\\_ATTR\\_DOMAINID](#) 8
- #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
- #define [LBMMON\\_RCTL\\_RECEIVER\\_TOPIC\\_CALLBACK](#) 4
- #define [LBMMON\\_RCTL\\_WILDCARD\\_RECEIVER\\_CALLBACK](#) 5

## 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 `lbm_src_transport_stats_t` structure.*
- typedef int(\*) `lbmmon_evq_format_deserialize_t` (`lbm_event_queue_stats_t` \*Statistics, const char \*Source, size\_t Length, unsigned short ModuleID, void \*FormatClientData)  
*Function to deserialize a buffer into an `lbm_event_queue_stats_t` structure.*
- typedef int(\*) `lbmmon_ctx_format_deserialize_t` (`lbm_context_stats_t` \*Statistics, const char \*Source, size\_t Length, unsigned short ModuleID, void \*FormatClientData)  
*Function to deserialize a buffer into an `lbm_context_stats_t` structure.*
- typedef int(\*) `lbmmon_rcv_topic_format_serialize_t` (char \*Destination, size\_t \*Size, unsigned short \*ModuleID, const char \*Topic, `lbm_ulong_t` SourceCount, const `lbm_rcv_topic_stats_t` \*Sources, void \*FormatClientData)  
*Function to serialize receiver topic statistics.*
- typedef int(\*) `lbmmon_rcv_topic_format_deserialize_t` (size\_t \*Count, `lbm_rcv_topic_stats_t` \*Statistics, const char \*Source, size\_t Length, unsigned short ModuleID, void \*FormatClientData)  
*Function to deserialize a buffer into an `lbm_rcv_topic_stats_t` structure.*
- typedef int(\*) `lbmmon_wildcard_rcv_format_serialize_t` (char \*Destination, size\_t \*Size, unsigned short \*ModuleID, const `lbm_wildcard_rcv_stats_t` \*Statistics, void \*FormatClientData)  
*Function to serialize wildcard receiver statistics.*

- typedef int(\*) [lbmmon\\_wildcard\\_rcv\\_format\\_deserialize\\_t](#) ([lbm\\_wildcard\\_rcv\\_stats\\_t](#) \*Statistics, const char \*Source, size\_t Length, unsigned short ModuleID, void \*FormatClientData)  
*Function to deserialize a buffer into an [lbm\\_wildcard\\_rcv\\_stats\\_t](#) structure.*
- typedef int(\*) [lbmmon\\_format\\_finish\\_t](#) (void \*FormatClientData)  
*Function to finish format module processing.*
- typedef const char \*(\*) [lbmmon\\_format\\_errmsg\\_t](#) (void)  
*Function to return the last error message from a format module.*
- typedef [lbmmon\\_format\\_func\\_t\\_stct](#) [lbmmon\\_format\\_func\\_t](#)  
*Format module function pointer container.*
- typedef void(\*) [lbmmon\\_rcv\\_statistics\\_cb](#) (const void \*AttributeBlock, const [lbm\\_rcv\\_transport\\_stats\\_t](#) \*Statistics, void \*ClientData)  
*Client callback function to process a received receiver statistics packet.*
- typedef [lbmmon\\_rcv\\_statistics\\_func\\_t\\_stct](#) [lbmmon\\_rcv\\_statistics\\_func\\_t](#)  
*A structure that holds the callback information for receiver statistics.*
- typedef void(\*) [lbmmon\\_src\\_statistics\\_cb](#) (const void \*AttributeBlock, const [lbm\\_src\\_transport\\_stats\\_t](#) \*Statistics, void \*ClientData)  
*Client callback function to process a received source statistics packet.*
- typedef [lbmmon\\_src\\_statistics\\_func\\_t\\_stct](#) [lbmmon\\_src\\_statistics\\_func\\_t](#)  
*A structure that holds the callback information for source statistics.*
- typedef void(\*) [lbmmon\\_evq\\_statistics\\_cb](#) (const void \*AttributeBlock, const [lbm\\_event\\_queue\\_stats\\_t](#) \*Statistics, void \*ClientData)  
*Client callback function to process a received event queue statistics packet.*
- typedef [lbmmon\\_evq\\_statistics\\_func\\_t\\_stct](#) [lbmmon\\_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 [lbm\\_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 void(\*) [lbmmon\\_rcv\\_topic\\_statistics\\_cb](#) (const void \*AttributeBlock, const [lbm\\_rcv\\_topic\\_stats\\_t](#) \*Statistics, void \*ClientData)  
*For internal use only. Client callback function to process a received receiver topic statistics packet.*
- typedef [lbmmon\\_rcv\\_topic\\_statistics\\_func\\_t\\_stct](#) [lbmmon\\_rcv\\_topic\\_statistics\\_func\\_t](#)  
*For internal use only. A structure that holds the callback information for receiver topic statistics.*
- typedef void(\*) [lbmmon\\_wildcard\\_rcv\\_statistics\\_cb](#) (const void \*AttributeBlock, const [lbm\\_wildcard\\_rcv\\_stats\\_t](#) \*Statistics, void \*ClientData)  
*For internal use only. Client callback function to process a received wildcard receiver statistics packet.*
- typedef [lbmmon\\_wildcard\\_rcv\\_statistics\\_func\\_t\\_stct](#) [lbmmon\\_wildcard\\_rcv\\_statistics\\_func\\_t](#)  
*A structure that holds the callback information for wildcard receiver 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

- LBMEpDLL int `lbmmon_sctl_create` (`lbmmon_sctl_t **Control`, const `lbmmon_format_func_t *Format`, const void `*FormatOptions`, const `lbmmon_transport_func_t *Transport`, const void `*TransportOptions`)  
*Create an LBM Monitoring Source Controller.*
- LBMEpDLL int `lbmmon_rctl_attr_create` (`lbmmon_rctl_attr_t **Attributes`)  
*Create an LBM Monitoring Receive Controller attribute object.*
- LBMEpDLL int `lbmmon_rctl_attr_delete` (`lbmmon_rctl_attr_t *Attributes`)  
*Delete an LBM Monitoring Receive Controller attribute object.*
- LBMEpDLL int `lbmmon_rctl_attr_setopt` (`lbmmon_rctl_attr_t *Attributes`, int `Option`, void `*Value`, `size_t Length`)  
*Set an LBMMON receive controller attribute option value.*
- LBMEpDLL int `lbmmon_rctl_attr_getopt` (`lbmmon_rctl_attr_t *Attributes`, int `Option`, void `*Value`, `size_t *Length`)  
*Get an LBMMON receive controller attribute option value.*
- LBMEpDLL int `lbmmon_rctl_create` (`lbmmon_rctl_t **Control`, const `lbmmon_format_func_t *Format`, const void `*FormatOptions`, const `lbmmon_transport_func_t *Transport`, const void `*TransportOptions`, `lbmmon_rctl_attr_t *Attributes`, void `*ClientData`)  
*Create an LBM Monitoring Receive Controller.*
- LBMEpDLL int `lbmmon_context_monitor` (`lbmmon_sctl_t *Control`, `lbm_context_t *Context`, const char `*ApplicationSourceID`, unsigned int `Seconds`)  
*Register a context for monitoring.*
- LBMEpDLL int `lbmmon_context_unmonitor` (`lbmmon_sctl_t *Control`, `lbm_context_t *Context`)  
*Terminate monitoring for a context.*

- LBMEExpDLL int [lbmmon\\_src\\_monitor](#) (lbmmon\_sctl\_t \*Control, lbm\_src\_t \*Source, const char \*ApplicationSourceID, unsigned int Seconds)  
*Register a source for monitoring.*
- LBMEExpDLL int [lbmmon\\_src\\_unmonitor](#) (lbmmon\_sctl\_t \*Control, lbm\_src\_t \*Source)  
*Terminate monitoring for a source.*
- LBMEExpDLL int [lbmmon\\_rcv\\_monitor](#) (lbmmon\_sctl\_t \*Control, lbm\_rcv\_t \*Receiver, const char \*ApplicationSourceID, unsigned int Seconds)  
*Register a receiver for monitoring.*
- LBMEExpDLL int [lbmmon\\_rcv\\_unmonitor](#) (lbmmon\_sctl\_t \*Control, lbm\_rcv\_t \*Receiver)  
*Terminate monitoring for a receiver.*
- LBMEExpDLL int [lbmmon\\_evq\\_monitor](#) (lbmmon\_sctl\_t \*Control, lbm\_event\_queue\_t \*EventQueue, const char \*ApplicationSourceID, unsigned int Seconds)  
*Register an event queue for monitoring.*
- LBMEExpDLL int [lbmmon\\_evq\\_unmonitor](#) (lbmmon\_sctl\_t \*Control, lbm\_event\_queue\_t \*EventQueue)  
*Terminate monitoring for an event queue.*
- LBMEExpDLL int [lbmmon\\_sctl\\_destroy](#) (lbmmon\_sctl\_t \*Control)  
*Destroy a source monitoring controller.*
- LBMEExpDLL int [lbmmon\\_rctl\\_destroy](#) (lbmmon\_rctl\_t \*Control)  
*Destroy a statistics receive controller.*
- LBMEExpDLL int [lbmmon\\_sctl\\_sample](#) (lbmmon\_sctl\_t \*Control)  
*Gather statistics for on-demand objects.*
- LBMEExpDLL int [lbmmon\\_sctl\\_sample\\_ex](#) (lbmmon\_sctl\_t \*Control, const char \*ApplicationSourceID)  
*Extended gather statistics for on-demand objects.*
- LBMEExpDLL int [lbmmon\\_attr\\_get\\_ipv4sender](#) (const void \*AttributeBlock, lbm\_uint\_t \*Address)  
*Retrieve the IPV4 sender address attribute from the statistics attribute block.*

- LBMEExpDLL int [lbmmon\\_attr\\_get\\_timestamp](#) (const void \*AttributeBlock, time\_t \*Timestamp)  
*Retrieve the timestamp attribute from the statistics attribute block.*
- LBMEExpDLL int [lbmmon\\_attr\\_get\\_appsourceid](#) (const void \*AttributeBlock, char \*ApplicationSourceID, size\_t Length)  
*Retrieve the application source ID attribute from the statistics attribute block.*
- LBMEExpDLL int [lbmmon\\_attr\\_get\\_objectid](#) (const void \*AttributeBlock, lbm\_ulong\_t \*ObjectID)  
*Retrieve the object ID attribute from the statistics attribute block.*
- LBMEExpDLL int [lbmmon\\_attr\\_get\\_contextid](#) (const void \*AttributeBlock, lbm\_ulong\_t \*ContextID)  
*Retrieve the context ID attribute from the statistics attribute block.*
- LBMEExpDLL int [lbmmon\\_attr\\_get\\_processid](#) (const void \*AttributeBlock, lbm\_ulong\_t \*ProcessID)  
*Retrieve the process ID attribute from the statistics attribute block.*
- LBMEExpDLL int [lbmmon\\_attr\\_get\\_source](#) (const void \*AttributeBlock, lbm\_ulong\_t \*Source)  
*Retrieve the source attribute from the statistics attribute block.*
- LBMEExpDLL int [lbmmon\\_attr\\_get\\_ctxinst](#) (const void \*AttributeBlock, lbm\_uint8\_t \*ContextInstance, size\_t Length)  
*For internal use only. Retrieve the context instance attribute from the statistics attribute block.*
- LBMEExpDLL int [lbmmon\\_attr\\_get\\_domainid](#) (const void \*AttributeBlock, lbm\_uint32\_t \*DomainID)  
*For internal use only. Retrieve the domain ID attribute from the statistics attribute block.*
- LBMEExpDLL int [lbmmon\\_errnum](#) (void)  
*Retrieve the error number for the last error encountered.*
- LBMEExpDLL const char \* [lbmmon\\_errmsg](#) (void)  
*Retrieve the error message for the last error encountered.*
- const char \* [lbmmon\\_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

**Version:**

```
//UMprod/REL_6_7_1/29West/lbm/src/mon/lbm/lbmmon.h#1
```

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\_CTXINST 7

Attribute block entry contains the context instance.

#### 8.4.2.4 #define LBMMON\_ATTR\_DOMAINID 8

Attribute block entry contains the domain ID.

#### 8.4.2.5 #define LBMMON\_ATTR\_FORMAT\_MODULEID 3

Attribute block entry contains the format module ID.

#### 8.4.2.6 #define LBMMON\_ATTR\_OBJECTID 4

Attribute block contains the object ID.

#### 8.4.2.7 #define LBMMON\_ATTR\_PROCESSID 5

Attribute block entry contains the process ID.

#### 8.4.2.8 #define LBMMON\_ATTR\_SENDER\_IPV4 0

Attribute block entry contains the sender IPV4 address.

#### 8.4.2.9 #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.

#### 8.4.2.10 #define LBMMON\_ATTR\_SOURCE\_IM 1

Source/receiver statistics are from a MIM transport session.

**8.4.2.11 #define LBMMON\_ATTR\_SOURCE\_NORMAL 0**

Source/receiver statistics are from a normal transport session.

**8.4.2.12 #define LBMMON\_ATTR\_TIMESTAMP 1**

Attribute block entry contains the timestamp.

**8.4.2.13 #define LBMMON\_EAGAIN (LBMMON\_ERROR\_BASE + 5)**

[lbmmon\\_errnum\(\)](#) value. Insufficient resources.

**8.4.2.14 #define LBMMON\_EALREADY (LBMMON\_ERROR\_BASE + 6)**

[lbmmon\\_errnum\(\)](#) value. Resource already registered.

**8.4.2.15 #define LBMMON\_EINVAL (LBMMON\_ERROR\_BASE + 1)**

[lbmmon\\_errnum\(\)](#) value. An invalid argument was passed.

**8.4.2.16 #define LBMMON\_ELBMFAIL (LBMMON\_ERROR\_BASE + 4)**

[lbmmon\\_errnum\(\)](#) value. A call to an LBM function failed.

**8.4.2.17 #define LBMMON\_EMODFAIL (LBMMON\_ERROR\_BASE + 3)**

[lbmmon\\_errnum\(\)](#) value. A call to a module function failed.

**8.4.2.18 #define LBMMON\_ENOMEM (LBMMON\_ERROR\_BASE + 2)**

[lbmmon\\_errnum\(\)](#) value. Out of memory.

**8.4.2.19 #define LBMMON\_EOP (LBMMON\_ERROR\_BASE + 7)**

[lbmmon\\_errnum\(\)](#) value. Internal operation error.

**8.4.2.20 #define LBMMON\_ERROR\_BASE 4096**

Base value for LBMMON error codes.

**8.4.2.21 #define LBMMON\_PACKET\_SIGNATURE 0x1b33041b**

Packet signature value

**8.4.2.22 #define LBMMON\_PACKET\_TYPE\_CONTEXT 3**

Packet contains context statistics

**8.4.2.23 #define LBMMON\_PACKET\_TYPE\_EVENT\_QUEUE 2**

Packet contains event queue statistics

**8.4.2.24 #define LBMMON\_PACKET\_TYPE\_RECEIVER 1**

Packet contains receiver statistics

**8.4.2.25 #define LBMMON\_PACKET\_TYPE\_RECEIVER\_TOPIC 4**

Packet contains receiver topic statistics.

**8.4.2.26 #define LBMMON\_PACKET\_TYPE\_SOURCE 0**

Packet contains source statistics

**8.4.2.27 #define LBMMON\_PACKET\_TYPE\_WILDCARD\_RECEIVER 5**

Packet contains wildcard receiver statistics.

**8.4.2.28 #define LBMMON\_RCTL\_CONTEXT\_CALLBACK 3**

Receive controller attribute option: Context statistics callback.

**8.4.2.29 #define LBMMON\_RCTL\_EVENT\_QUEUE\_CALLBACK 2**

Receive controller attribute option: Event queue statistics callback.

**8.4.2.30 #define LBMMON\_RCTL\_RECEIVER\_CALLBACK 0**

Receive controller attribute option: Receiver statistics callback.

#### 8.4.2.31 `#define LBMMON_RCTL_RECEIVER_TOPIC_CALLBACK 4`

Receive controller attribute option: Receiver topic statistics callback.

#### 8.4.2.32 `#define LBMMON_RCTL_SOURCE_CALLBACK 1`

Receive controller attribute option: Source statistics callback.

#### 8.4.2.33 `#define LBMMON_RCTL_WILDCARD_RECEIVER_CALLBACK 5`

Receive controller attribute option: Wildcard receiver statistics callback.

### 8.4.3 Typedef Documentation

#### 8.4.3.1 `typedef int(*) lbmmon_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 `lbmmon_ctx_format_serialize_t` function into a `lbm_context_stats_t` structure.

#### See also:

[lbmmon\\_ctx\\_format\\_serialize\\_t](#)

#### Parameters:

*Statistics* A pointer to an `lbm_context_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.

**8.4.3.2** `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(*) lbmmon_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 `lbmmon_evq_format_serialize_t` function into a `lbm_event_queue_stats_t` structure.

**See also:**

[lbmmon\\_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 `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.

#### 8.4.3.6 `typedef int(*) lbmmon_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 `lbmmon_evq_format_deserialize_t` function.

**See also:**

[lbmmon\\_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 `lbmmon_format_init_t` function.

**Returns:**

Zero if successful, -1 otherwise. If -1 is returned, the serialized data will not be sent.

**8.4.3.7** `typedef void(*) lbmmon_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 `lbmmon_rctl_create()`.

**8.4.3.8** `typedef struct lbmmon_evq_statistics_func_t_stct lbmmon_evq_statistics_func_t`

A structure used with receive controller options to get/set specific callback information.

**8.4.3.9** `typedef const char*(*) lbmmon_format_errmsg_t(void)`

**Returns:**

A string containing a description of the last error encountered by the module.

**8.4.3.10** `typedef int(*) lbmmon_format_finish_t(void *FormatClientData)`

Perform any required format module cleanup processing. If format-specific client data was allocated in the `lbmmon_format_init_t` function, it should be freed in this function.

This function is called by `lbmmon_sctl_destroy()` and `lbmmon_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(*) lbmmon_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 [lbmmon\\_sctl\\_create\(\)](#) and [lbmmon\\_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 [lbmmon\\_sctl\\_create\(\)](#) or [lbmmon\\_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 ([lbmmon\\_sctl\\_create\(\)](#) or [lbmmon\\_rctl\\_create\(\)](#)) will return -1.

**8.4.3.12** `typedef struct lbmmon_packet_hdr_t_stct lbmmon_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(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.

**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 `lbm_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 `lbm_rcv_transport_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.

**8.4.3.15** `typedef void(*) lbmmon_rcv_statistics_cb(const void *AttributeBlock, const lbm_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 `lbmmon_rctl_create()`.

**8.4.3.16** `typedef struct lbmmon_rcv_statistics_func_t_stct lbmmon_rcv_statistics_func_t`

A structure used with receive controller options to get/set specific callback information.

**8.4.3.17** `typedef int(*) lbmmon_rcv_topic_format_deserialize_t(size_t *Count, lbm_rcv_topic_stats_t *Statistics, const char *Source, size_t Length, unsigned short ModuleID, void *FormatClientData)`

Transform a block of data serialized by the `lbmmon_rcv_topic_format_serialize_t` function into a `lbm_rcv_topic_stats_t` structure.

**See also:**

[lbmmon\\_rcv\\_topic\\_format\\_deserialize\\_t](#)

**Parameters:**

**Count** A pointer to an integer containing the number of elements in the *Statistics* array. On exit, it will contain the actual number of elements parsed.

**Statistics** An array of [lbm\\_rcv\\_topic\\_stats\\_t](#) elements into which is written the actual deserialized data.

**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, -2 if *Count* is not large enough for all elements, -1 otherwise. If -2 is returned, *Count* will contain the number of elements required. If -1 is returned, the deserialized data will not be delivered to the application.

**8.4.3.18** `typedef int(*) lbmmon_rcv_topic_format_serialize_t(char *Destination, size_t *Size, unsigned short *ModuleID, const char *Topic, lbm_ulong_t SourceCount, const lbm_rcv_topic_stats_t *Sources, void *FormatClientData)`

This function should transform the topic, source count, and array of sources into a form which can be deserialized by the corresponding [lbmmon\\_rcv\\_topic\\_format\\_deserialize\\_t](#) function.

**See also:**

[lbmmon\\_rcv\\_topic\\_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).

**Topic** A NUL-terminated string containing the topic.

**SourceCount** The number of sources in the *Sources* array.

**Sources** An array of [lbm\\_rcv\\_topic\\_stats\\_t](#) structures containing the sources to which the receiver is joined.

**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.

**8.4.3.19** `typedef void(*) lbmmon_rcv_topic_statistics_cb(const void *AttributeBlock, const lbm_rcv_topic_stats_t *Statistics, void *ClientData)`

#### Parameters:

**AttributeBlock** Pointer to the statistics packet attribute block.

**Statistics** Pointer to the receiver topic statistics.

**ClientData** Pointer to client-specific data as passed to [lbmmon\\_rctl\\_create\(\)](#).

**8.4.3.20** `typedef struct lbmmon_rcv_topic_statistics_func_t_stct lbmmon_rcv_topic_statistics_func_t`

A structure used with receive controller options to get/set specific callback information.

**8.4.3.21** `typedef int(*) lbmmon_src_format_deserialize_t(lbm_src_transport_stats_t *Statistics, const char *Source, size_t Length, unsigned short ModuleID, void *FormatClientData)`

#### See also:

[lbmmon\\_rcv\\_format\\_serialize\\_t](#)

[lbmmon\\_src\\_format\\_serialize\\_t](#)

[lbmmon\\_rcv\\_format\\_deserialize\\_t](#)

**Parameters:**

**Statistics** A pointer to an `lbm_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 `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.

**8.4.3.22** `typedef int(*) lbmmon_src_format_serialize_t(char *Destination, size_t *Size, unsigned short *ModuleID, const lbm_src_transport_stats_t *Statistics, void *FormatClientData)`

**See also:**

[lbmmon\\_rcv\\_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 `lbm_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 `lbm_src_transport_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.

**8.4.3.23** `typedef void(*) lbmmon_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 `lbmmon_rctl_create()`.

**8.4.3.24** `typedef struct lbmmon_src_statistics_func_t_stct lbmmon_src_statistics_func_t`

A structure used with receive controller options to get/set specific callback information.

**8.4.3.25** `typedef const char*(*) lbmmon_transport_errmsg_t(void)`

**Returns:**

A string containing a description of the last error encountered by the module.

**8.4.3.26** `typedef int(*) lbmmon_transport_finishrv_t(void *TransportClientData)`

**Parameters:**

*TransportClientData* A pointer to transport-specific client data as returned by the `lbmmon_transport_initrv_t` function. If previously allocated by the `lbmmon_transport_initrv_t` function, it must be freed in this function.

**Returns:**

Zero if successful, -1 otherwise.

**8.4.3.27** `typedef int(*) lbmmon_transport_finishsrc_t(void *TransportClientData)`

**Parameters:**

*TransportClientData* A pointer to transport-specific client data as returned by the `lbmmon_transport_initsrc_t` function. If previously allocated by the `lbmmon_transport_initsrc_t` function, it must be freed in this function.

**Returns:**

Zero if successful, -1 otherwise.

**8.4.3.28** `typedef int(*) lbmmon_transport_initrv_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.

**8.4.3.29** `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.

**8.4.3.30** `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\\_initrv\\_t](#) function.

**Returns:**

Zero if successful, >0 if timeout exceeded, -1 otherwise.

**8.4.3.31** `typedef int(*) lbmmon_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 [lbmmon\\_transport\\_initsrc\\_t](#) function.

**Returns:**

Zero if successful, -1 otherwise.

**8.4.3.32** `typedef int(*) lbmmon_wildcard_rcv_format_deserialize_t(lbm_wildcard_rcv_stats_t *Statistics, const char *Source, size_t Length, unsigned short ModuleID, void *FormatClientData)`

Transform a block of data serialized by the [lbmmon\\_wildcard\\_rcv\\_format\\_serialize\\_t](#) function into a [lbm\\_wildcard\\_rcv\\_stats\\_t](#) structure.

**See also:**

[lbmmon\\_wildcard\\_rcv\\_format\\_deserialize\\_t](#)

**Parameters:**

*Statistics* A pointer to an [lbm\\_wildcard\\_rcv\\_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.

**8.4.3.33** `typedef int(*) lbmmon_wildcard_rcv_format_serialize_t(char *Destination, size_t *Size, unsigned short *ModuleID, const lbm_wildcard_rcv_stats_t *Statistics, void *FormatClientData)`

This function should transform the `lbm_wildcard_receiver_stats_t` structure into a form which can be deserialized by the corresponding `lbmmon_wildcard_rcv_format_deserialize` function.

**See also:**

[lbmmon\\_wildcard\\_rcv\\_format\\_deserialize\\_t](#)

**Parameters:**

*Destination* A pointer to a buffer to receive the serialized format of the `lbm_wildcard_rcv_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 an `lbm_wildcard_rcv_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.

**8.4.3.34** `typedef void(*) lbmmon_wildcard_rcv_statistics_cb(const void *AttributeBlock, const lbm_wildcard_rcv_stats_t *Statistics, void *ClientData)`

**Parameters:**

*AttributeBlock* Pointer to the statistics packet attribute block.

*Statistics* Pointer to the wildcard receiver statistics.

*ClientData* Pointer to client-specific data as passed to `lbmmon_rctl_create()`.

#### 8.4.3.35 `typedef struct lbmmon_wildcard_rcv_statistics_func_t_stct` `lbmmon_wildcard_rcv_statistics_func_t`

A structure used with receive controller options to get/set specific callback information.

### 8.4.4 Function Documentation

#### 8.4.4.1 `LBMEExpDLL int lbmmon_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 lbmmon_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 `lbmmon_attr_get_objectid` instead.

#### 8.4.4.3 `LBMEExpDLL int lbmmon_attr_get_ctxinst (const void * AttributeBlock,` `lbm_uint8_t * ContextInstance, size_t Length)`

**Parameters:**

*AttributeBlock* Pointer to the attribute block as passed to the callback function.

*ContextInstance* Pointer to a buffer to receive the context instance.

*Length* Maximum length of *ContextInstance*.

**Returns:**

0 if successful, -1 if the attribute does not exist.

**8.4.4.4 LBMEpDLL int lbmmon\_attr\_get\_domainid (const void \*  
*AttributeBlock*, lbm\_uint32\_t \* *DomainID*)****Parameters:**

*AttributeBlock* Pointer to the attribute block as passed to the callback function.

*DomainID* Pointer to a variable to receive the domain ID.

**Returns:**

0 if successful, -1 if the attribute does not exist.

**8.4.4.5 LBMEpDLL int lbmmon\_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.

**8.4.4.6 LBMEpDLL int lbmmon\_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.

#### 8.4.4.7 LBMExpDLL int lbmmon\_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.

#### 8.4.4.8 LBMExpDLL int lbmmon\_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.

#### 8.4.4.9 LBMExpDLL int lbmmon\_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.

#### 8.4.4.10 LBMExpDLL int lbmmon\_context\_monitor (lbmmon\_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 [lbmmon\\_sctl\\_sample\(\)](#).

If monitoring is to be used as a form of heartbeat, the preferred method is to call `lbmmon_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 `lbmmon_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.

**8.4.4.11 LBMEpDLL 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.

**8.4.4.12 LBMEpDLL const char\* lbmmon\_errmsg (void)****Returns:**

A pointer to a static character array containing the last error message.

**8.4.4.13 LBMEpDLL int lbmmon\_errnum (void)****Returns:**

The last error encountered. See `LBMMON_ERR_*`.

#### 8.4.4.14 LBMEpDLL 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 `lbmmon_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 `lbmmon_sctl_t` which is to be used to monitor the context.

*Receiver* Pointer to an `lbm_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 `lbmmon_sctl_sample()`.

##### Returns:

Zero if successful, -1 otherwise.

#### 8.4.4.15 LBMEpDLL int lbmmon\_evq\_unmonitor (lbmmon\_sctl\_t \* *Control*, lbm\_event\_queue\_t \* *EventQueue*)

Unregister an event queue to prevent further monitoring of that receiver.

##### Parameters:

*Control* Pointer to an `lbmmon_sctl_t` which is used to monitor the context.

*EventQueue* Pointer to an previously registered `lbm_event_queue_t`.

##### Returns:

Zero if successful, -1 otherwise.

**8.4.4.16** `const char* lbmmon_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 [lbmmon\\_next\\_key\\_value\\_pair\(\)](#).

**8.4.4.17** `LBMEpDLL int lbmmon_rctl_attr_create (lbmmon_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 `lbmmon_rctl_attr_t` object.

**Returns:**

0 if successful, -1 otherwise.

**8.4.4.18** `LBMEpDLL int lbmmon_rctl_attr_delete (lbmmon_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 [lbmmon\\_rctl\\_attr\\_create](#).

**Returns:**

0 if successful, -1 otherwise.

#### 8.4.4.19 LBMEpDLL int lbmmon\_rctl\_attr\_getopt (lbmmon\_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.

#### 8.4.4.20 LBMEpDLL int lbmmon\_rctl\_attr\_setopt (lbmmon\_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.

#### 8.4.4.21 LBMEpDLL int lbmmon\_rctl\_create (lbmmon\_rctl\_t \*\* *Control*, const lbmmon\_format\_func\_t \* *Format*, const void \* *FormatOptions*, const lbmmon\_transport\_func\_t \* *Transport*, const void \* *TransportOptions*, lbmmon\_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 lbmmon\_rctl\_t object.

*Format* A pointer to an lbmmon\_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 `lbmmon_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 `lbmmon_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.

#### 8.4.4.22 LBMEExpDLL int lbmmon\_rctl\_destroy (lbmmon\_rctl\_t \* *Control*)

Destroys a monitoring controller.

**Parameters:**

**Control** Pointer to an `lbmmon_rctl_t` to be destroyed.

**Returns:**

Zero if successful, -1 otherwise.

#### 8.4.4.23 LBMEExpDLL int lbmmon\_rcv\_monitor (lbmmon\_sctl\_t \* *Control*, lbm\_rcv\_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 `lbmmon_ctl_sample()`.

If monitoring is to be used as a form of heartbeat, the preferred method is to call `lbmmon_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 `lbmmon_sctl_t` which is to be used to monitor the context.

**Receiver** Pointer to an `lbm_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 `lbmmon_sctl_sample()`.

**Returns:**

Zero if successful, -1 otherwise.

**8.4.4.24 LBMEpDLL int lbmmon\_rcv\_unmonitor (lbmmon\_sctl\_t \* Control, lbm\_rcv\_t \* Receiver)**

Unregister a receiver to prevent further monitoring of that receiver.

**Parameters:**

**Control** Pointer to an `lbmmon_sctl_t` which is used to monitor the context.

**Receiver** Pointer to an previously registered `lbm_rcv_t`.

**Returns:**

Zero if successful, -1 otherwise.

**8.4.4.25 LBMEpDLL int lbmmon\_sctl\_create (lbmmon\_sctl\_t \*\* Control, const lbmmon\_format\_func\_t \* Format, const void \* FormatOptions, const lbmmon\_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 `lbmmon_sctl_t` object.

**Format** A pointer to an `lbmmon_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 `lbmmon_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.

**8.4.4.26 LBMEExpDLL int lbmmon\_sctl\_destroy (lbmmon\_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 `lbmmon_sctl_t` to be destroyed.

**Returns:**

Zero if successful, -1 otherwise.

**8.4.4.27 LBMEExpDLL int lbmmon\_sctl\_sample (lbmmon\_sctl\_t \* Control)**

**Parameters:**

**Control** Pointer to an existing `lbmmon_sctl_t` controller.

**Returns:**

Zero if successful, -1 otherwise.

**8.4.4.28 LBMEExpDLL int lbmmon\_sctl\_sample\_ex (lbmmon\_sctl\_t \* Control, const char \* ApplicationSourceID)**

**Parameters:**

**Control** Pointer to an existing `lbmmon_sctl_t` controller.

**ApplicationSourceID** Null-terminated string containing an application-specified source identifier. This overrides any application source ID passed to any of the `lbmmon_*_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.

#### 8.4.4.29 LBMEpDLL int lbmmon\_src\_monitor (lbmmon\_sctl\_t \* *Control*, lbm\_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 [lbmmon\\_sctl\\_sample\(\)](#).

If monitoring is to be used as a form of heartbeat, the preferred method is to call [lbmmon\\_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 `lbmmon_sctl_t` which is to be used to monitor the context.

*Source* Pointer to an `lbm_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 [lbmmon\\_sctl\\_sample\(\)](#).

##### Returns:

Zero if successful, -1 otherwise.

#### 8.4.4.30 LBMEpDLL int lbmmon\_src\_unregister (lbmmon\_sctl\_t \* *Control*, lbm\_src\_t \* *Source*)

Unregister a source to prevent further monitoring of that source.

##### Parameters:

*Control* Pointer to an `lbmmon_sctl_t` which is used to monitor the context.

*Source* Pointer to an previously registered `lbm_src_t`.

##### Returns:

Zero if successful, -1 otherwise.

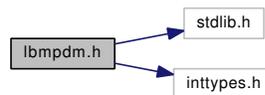
## 8.5 lbmpdm.h File Reference

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 [LBMPDMExpDLL](#)  
*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 `lbmpdm_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

- LBMPDMExpDLL int [lbmpdm\\_errnum](#) ()  
*Return the error number last encountered by this thread.*
- LBMPDMExpDLL const char \* [lbmpdm\\_errmsg](#) ()  
*Return an ASCII string containing the error message last encountered by this thread.*
- LBMPDMExpDLL 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.*
- LBMPDMExpDLL 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.*
- LBMPDMExpDLL int [lbmpdm\\_cache\\_struct\\_remove](#) (int32\_t id)  
*delete a definition structure from the cache. Does not error if the structure doesn't exist.*
- LBMPDMExpDLL 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.*
- LBMPDMExpDLL 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.*
- LBMPDMExpDLL 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.*
- LBMPDMExpDLL int [lbmpdm\\_cache\\_clear\\_all](#) ()  
*nuke the whole cache, this deletes all the structures within the cache as well.*
- LBMPDMExpDLL [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.*

- LBMPDMExpDLL `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.*

- 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)`

*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.*

- LBMPDMExpDLL `int lbmpdm_defn_delete (lbmpdm_defn_t *defn)`

*delete a given definition.*

- LBMPDMExpDLL `int lbmpdm_defn_finalize (lbmpdm_defn_t *defn)`

*make this definition final. This needs to be done before using it in a message.*

- 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)`

*adds field info to the definition by string name*

- 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)`

*adds field info to the definition by integer name*

- LBMPDMExpDLL `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.*

- LBMPDMExpDLL `int32_t lbmpdm_defn_get_id (lbmpdm_defn_t *defn)`

*Gets the id of the definition.*

- LBMPDMExpDLL `int32_t lbmpdm_defn_get_num_fields (lbmpdm_defn_t *defn)`

*Gets the number of fields in the definition.*

- LBMPDMExpDLL `int8_t lbmpdm_defn_get_msg_vers_major (lbmpdm_defn_t *defn)`

*Gets the message major version number from the definition.*

- `LBMPDMExpDLL int8_t lbmpdm_defn_get_msg_vers_minor (lbmpdm_defn_t *defn)`  
*Gets the message minor version number from the definition.*
- `LBMPDMExpDLL 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.*
- `LBMPDMExpDLL 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`).*
- `LBMPDMExpDLL 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.*
- `LBMPDMExpDLL 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.*
- `LBMPDMExpDLL 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.*
- `LBMPDMExpDLL 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.*
- `LBMPDMExpDLL 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.*
- `LBMPDMExpDLL int lbmpdm_msg_create (lbmpdm_msg_t **message, lbmpdm_defn_t *defn, uint32_t flags)`  
*creates a message with the specified definition*
- `LBMPDMExpDLL 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.

- LBMPDMExpDLL 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.
- LBMPDMExpDLL 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.
- LBMPDMExpDLL `lbmpdm_defn_t *``lbmpdm_msg_get_defn` (const `lbmpdm_msg_t *message`)  
Gets a pointer to the message definition.
- LBMPDMExpDLL 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.
- LBMPDMExpDLL 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.
- LBMPDMExpDLL 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.
- LBMPDMExpDLL 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.
- LBMPDMExpDLL 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.
- LBMPDMExpDLL 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.*

- LBMPDMEpDLL 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).*

- LBMPDMEpDLL int `lbmpdm_msg_set_incl_defn_flag` (`lbmpdm_msg_t *message`)

*Sets the message include definition flag.*

- LBMPDMEpDLL int `lbmpdm_msg_unset_incl_defn_flag` (`lbmpdm_msg_t *message`)

*Unsets the message include definition flag.*

- LBMPDMEpDLL 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.*

- LBMPDMEpDLL 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.*

- LBMPDMEpDLL int `lbmpdm_msg_deserialize` (`lbmpdm_msg_t *message`, `const char *bufptr`, `uint32_t buflen`)

*Deserialize the associated buffer into a newly created message.*

- LBMPDMEpDLL 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.*

- LBMPDMEpDLL int `lbmpdm_iter_create` (`lbmpdm_iter_t **iter`, `lbmpdm_msg_t *message`)

*Creates a pdm iterator to iterate through the fields in a message.*

- LBMPDMEpDLL 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.*

- LBMPDMExpDLL int `lbmpdm_iter_delete` (`lbmpdm_iter_t *iter`)  
*Deletes the iterator.*
  
- LBMPDMExpDLL `lbmpdm_field_handle_t` `lbmpdm_iter_get_current` (`lbmpdm_iter_t *iter`)  
*Gets the current field handle from the iterator.*
  
- LBMPDMExpDLL int `lbmpdm_iter_first` (`lbmpdm_iter_t *iter`)  
*Sets the iterator back to the first field.*
  
- LBMPDMExpDLL int `lbmpdm_iter_next` (`lbmpdm_iter_t *iter`)  
*Steps the iterator to the next first field.*
  
- LBMPDMExpDLL uint8\_t `lbmpdm_iter_has_next` (`lbmpdm_iter_t *iter`)  
*Checks to see if the iterator has another field to step to.*
  
- LBMPDMExpDLL uint8\_t `lbmpdm_iter_is_current_set` (`lbmpdm_iter_t *iter`)  
*Checks to see if the current field is set.*
  
- LBMPDMExpDLL int `lbmpdm_iter_set_msg` (`lbmpdm_iter_t *iter`, `lbmpdm_msg_t *message`)  
*Sets the message used to step through by this iterator.*
  
- LBMPDMExpDLL 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.*
  
- LBMPDMExpDLL 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.*
  
- LBMPDMExpDLL 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.*
  
- LBMPDMExpDLL 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 LBMPDMEpDLL 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 LBMPDMEpDLL int lbmpdm\_cache\_struct\_add (lbmpdm\_defn\_t \* *defn*)

**Parameters:**

*defn* – the new definition structure being added.

### 8.5.3.3 LBMPDMEpDLL 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 LBMPDMEpDLL 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 LBMPDMEpDLL 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 LBMPDMEpDLL 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 deserializable 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 LBMPDMExpDLL int lbmpdm\_defn\_delete ([lbmpdm\\_defn\\_t](#) \* *defn*)

**Parameters:**

*defn* – definition to be deleted.

**Returns:**

PDM\_SUCCESS or PDM\_FAILURE.

### 8.5.3.11 LBMPDMExpDLL 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 LBMPDMExpDLL 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 LBMPDMExpDLL 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 LBMPDMExpDLL 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 LBMPDMExpDLL 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 LBMPDMExpDLL 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 LBMPDMExpDLL 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 LBMPDMExpDLL 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** LBMPDMExpDLL 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** LBMPDMExpDLL 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** LBMPDMExpDLL 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** LBMPDMExpDLL 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** LBMPDMExpDLL 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** LBMPDMExpDLL 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** LBMPDMExpDLL 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 LBMPDMExpDLL 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 LBMPDMExpDLL 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 LBMPDMExpDLL 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 LBMPDMExpDLL 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 LBMPDMExpDLL 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 LBMPDMExpDLL 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 LBMPDMExpDLL 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 deserialized

*buflen* The length of the buffer.

**Returns:**

PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

#### 8.5.3.46 LBMPDMExpDLL 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 LBMPDMExpDLL `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** LBMPDMExpDLL 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** LBMPDMExpDLL 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** LBMPDMExpDLL 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 LBMPDMExpDLL 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 LBMPDMExpDLL 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 LBMPDMExpDLL 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 LBMPDMExpDLL 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 LBMPDMExpDLL 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 LBMPDMExpDLL 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:**

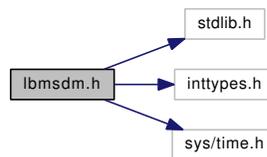
PDM\_SUCCESS if successful or PDM\_FAILURE otherwise.

## 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

- LBMSDMExpDLL int `lbmsdm_errnum` (void)
 

*Return the error number last encountered by this thread.*
- LBMSDMExpDLL const char \* `lbmsdm_errmsg` (void)
 

*Return an ASCII string containing the error message last encountered by this thread.*
- LBMSDMExpDLL 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.

- LBMSDMEExpDLL int [lbmsdm\\_msg\\_create](#) ([lbmsdm\\_msg\\_t](#) \*\*Message)  
*Create an SDM message to be filled in and sent.*
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_create\\_ex](#) ([lbmsdm\\_msg\\_t](#) \*\*Message, const [lbmsdm\\_msg\\_attr\\_t](#) \*Attributes)  
*Create an SDM message to be filled in and sent, with options.*
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_parse\\_ex](#) ([lbmsdm\\_msg\\_t](#) \*\*Message, const char \*Data, size\_t Length, const [lbmsdm\\_msg\\_attr\\_t](#) \*Attributes)  
*Create an SDM message to be parsed and processed from an existing buffer, with options.*
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_clone](#) ([lbmsdm\\_msg\\_t](#) \*\*Message, const [lbmsdm\\_msg\\_t](#) \*Original)  
*Clone an existing SDM message. This function is not thread safe.*
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_clear](#) ([lbmsdm\\_msg\\_t](#) \*Message)  
*Clear an SDM message, deleting all fields in the message.*
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_destroy](#) ([lbmsdm\\_msg\\_t](#) \*Message)  
*Destroy an SDM message object.*
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_dump](#) ([lbmsdm\\_msg\\_t](#) \*Message, char \*Buffer, size\_t Size)  
*Dump a message into a printable string.*
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_boolean](#) ([lbmsdm\\_msg\\_t](#) \*Message, const char \*Name, uint8\_t Value)  
*Add a field to a message.*

- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_int8](#) (lbmsdm\_msg\_t \*Message, const char \*Name, int8\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_uint8](#) (lbmsdm\_msg\_t \*Message, const char \*Name, uint8\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_int16](#) (lbmsdm\_msg\_t \*Message, const char \*Name, int16\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_uint16](#) (lbmsdm\_msg\_t \*Message, const char \*Name, uint16\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_int32](#) (lbmsdm\_msg\_t \*Message, const char \*Name, int32\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_uint32](#) (lbmsdm\_msg\_t \*Message, const char \*Name, uint32\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_int64](#) (lbmsdm\_msg\_t \*Message, const char \*Name, int64\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_uint64](#) (lbmsdm\_msg\_t \*Message, const char \*Name, uint64\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_float](#) (lbmsdm\_msg\_t \*Message, const char \*Name, float Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_double](#) (lbmsdm\_msg\_t \*Message, const char \*Name, double Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_decimal](#) (lbmsdm\_msg\_t \*Message, const char \*Name, const [lbmsdm\\_decimal\\_t](#) \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_timestamp](#) (lbmsdm\_msg\_t \*Message, const char \*Name, const struct timeval \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_message](#) (lbmsdm\_msg\_t \*Message, const char \*Name, const [lbmsdm\\_msg\\_t](#) \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_string](#) (lbmsdm\_msg\_t \*Message, const char \*Name, const char \*Value)
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_boolean\\_array](#) (lbmsdm\_msg\_t \*Message, const char \*Name)  
*Add an array field to a message.*
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_int8\\_array](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_uint8\\_array](#) (lbmsdm\_msg\_t \*Message, const char \*Name)

- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_int16\\_array](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_uint16\\_array](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_int32\\_array](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_uint32\\_array](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_int64\\_array](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_uint64\\_array](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_float\\_array](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_double\\_array](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_decimal\\_array](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_timestamp\\_array](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_message\\_array](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_string\\_array](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_unicode\\_array](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_blob\\_array](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEpDLL 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.*

- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_int8\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, int8\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_uint8\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, uint8\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_int16\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, int16\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_uint16\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, uint16\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_int32\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, int32\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_add\\_uint32\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, uint32\_t Value)

- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_int64\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, int64\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_uint64\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, uint64\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_float\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, float Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_double\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, double Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_decimal\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, const lbmsdm\_decimal\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_timestamp\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, const struct timeval \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_message\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, const lbmsdm\_msg\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_string\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, const char \*Value)
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_int8\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, int8\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_uint8\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, uint8\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_int16\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, int16\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_uint16\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, uint16\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_int32\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, int32\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_uint32\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, uint32\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_int64\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, int64\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_add\\_uint64\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, uint64\_t Value)

- LBMSDMEpDLL int `lbmsdm_msg_add_float_elem_name` (`lbmsdm_msg_t *Message`, const char \*Name, float Value)
- LBMSDMEpDLL int `lbmsdm_msg_add_double_elem_name` (`lbmsdm_msg_t *Message`, const char \*Name, double Value)
- LBMSDMEpDLL int `lbmsdm_msg_add_decimal_elem_name` (`lbmsdm_msg_t *Message`, const char \*Name, const `lbmsdm_decimal_t *Value`)
- LBMSDMEpDLL int `lbmsdm_msg_add_timestamp_elem_name` (`lbmsdm_msg_t *Message`, const char \*Name, const struct timeval \*Value)
- LBMSDMEpDLL int `lbmsdm_msg_add_message_elem_name` (`lbmsdm_msg_t *Message`, const char \*Name, const `lbmsdm_msg_t *Value`)
- LBMSDMEpDLL int `lbmsdm_msg_add_string_elem_name` (`lbmsdm_msg_t *Message`, const char \*Name, const char \*Value)
- LBMSDMEpDLL 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.*

- LBMSDMEpDLL 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.*

- LBMSDMEpDLL 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.*

- LBMSDMEpDLL int `lbmsdm_iter_add_int8_elem` (`lbmsdm_iter_t *Iterator`, `int8_t Value`)
- LBMSDMEpDLL int `lbmsdm_iter_add_uint8_elem` (`lbmsdm_iter_t *Iterator`, `uint8_t Value`)
- LBMSDMEpDLL int `lbmsdm_iter_add_int16_elem` (`lbmsdm_iter_t *Iterator`, `int16_t Value`)
- LBMSDMEpDLL int `lbmsdm_iter_add_uint16_elem` (`lbmsdm_iter_t *Iterator`, `uint16_t Value`)
- LBMSDMEpDLL int `lbmsdm_iter_add_int32_elem` (`lbmsdm_iter_t *Iterator`, `int32_t Value`)
- LBMSDMEpDLL int `lbmsdm_iter_add_uint32_elem` (`lbmsdm_iter_t *Iterator`, `uint32_t Value`)
- LBMSDMEpDLL int `lbmsdm_iter_add_int64_elem` (`lbmsdm_iter_t *Iterator`, `int64_t Value`)
- LBMSDMEpDLL int `lbmsdm_iter_add_uint64_elem` (`lbmsdm_iter_t *Iterator`, `uint64_t Value`)
- LBMSDMEpDLL int `lbmsdm_iter_add_float_elem` (`lbmsdm_iter_t *Iterator`, float Value)
- LBMSDMEpDLL 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.*

- LBMSDMEExpDLL int `lbsdm_msg_get_idx_name` (`lbsdm_msg_t *Message, const char *Name`)

*Get the index of a field in a message by field name.*

- LBMSDMEExpDLL const char \* `lbsdm_iter_get_name` (`lbsdm_iter_t *Iterator`)

*Get the name of the current field for an iterator.*

- LBMSDMEExpDLL `lbsdm_field_type_t` `lbsdm_msg_get_type_name` (`lbsdm_msg_t *Message, const char *Name`)

*Get the type of a field in a message by field name.*

- LBMSDMEExpDLL `lbsdm_field_type_t` `lbsdm_msg_get_type_idx` (`lbsdm_msg_t *Message, size_t Index`)

*Get the type of a field in a message by field index.*

- LBMSDMEExpDLL `lbsdm_field_type_t` `lbsdm_iter_get_type` (`lbsdm_iter_t *Iterator`)

*Get the type of the current field for an iterator.*

- LBMSDMEExpDLL int `lbsdm_msg_is_null_name` (`lbsdm_msg_t *Message, const char *Name`)

*Determine if a field in a message is null, by field name.*

- LBMSDMEExpDLL int `lbsdm_msg_is_null_idx` (`lbsdm_msg_t *Message, size_t Index`)

*Determine if a field in a message is null, by field index.*

- LBMSDMEExpDLL int `lbsdm_iter_is_null` (`lbsdm_iter_t *Iterator`)

*Determine if the field referenced by an iterator is null.*

- LBMSDMEExpDLL int `lbsdm_msg_get_elemcnt_name` (`lbsdm_msg_t *Message, const char *Name`)

*Get the number of elements in an array field in a message by field name.*

- LBMSDMEExpDLL int `lbsdm_msg_get_elemcnt_idx` (`lbsdm_msg_t *Message, size_t Index`)

*Get the number of elements in an array field by field index.*

- LBMSDMEExpDLL int `lbsdm_iter_get_elemcnt` (`lbsdm_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)

- LBMSDMEpDLL int `lbmsdm_msg_get_int64_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, int64\_t \*Value)
- LBMSDMEpDLL int `lbmsdm_msg_get_uint64_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, uint64\_t \*Value)
- LBMSDMEpDLL int `lbmsdm_msg_get_float_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, float \*Value)
- LBMSDMEpDLL int `lbmsdm_msg_get_double_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, double \*Value)
- LBMSDMEpDLL int `lbmsdm_msg_get_decimal_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, `lbmsdm_decimal_t` \*Value)
- LBMSDMEpDLL int `lbmsdm_msg_get_timestamp_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, struct timeval \*Value)
- LBMSDMEpDLL int `lbmsdm_msg_get_message_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, `lbmsdm_msg_t` \*\*Value)
- LBMSDMEpDLL 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.*

- LBMSDMEpDLL 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.*

- LBMSDMEpDLL 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.*

- LBMSDMEpDLL 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.*

- LBMSDMEpDLL int `lbmsdm_msg_get_int8_name` (`lbmsdm_msg_t` \*Message, const char \*Name, int8\_t \*Value)
- LBMSDMEpDLL int `lbmsdm_msg_get_uint8_name` (`lbmsdm_msg_t` \*Message, const char \*Name, uint8\_t \*Value)
- LBMSDMEpDLL int `lbmsdm_msg_get_int16_name` (`lbmsdm_msg_t` \*Message, const char \*Name, int16\_t \*Value)
- LBMSDMEpDLL int `lbmsdm_msg_get_uint16_name` (`lbmsdm_msg_t` \*Message, const char \*Name, uint16\_t \*Value)
- LBMSDMEpDLL int `lbmsdm_msg_get_int32_name` (`lbmsdm_msg_t` \*Message, const char \*Name, int32\_t \*Value)
- LBMSDMEpDLL int `lbmsdm_msg_get_uint32_name` (`lbmsdm_msg_t` \*Message, const char \*Name, uint32\_t \*Value)
- LBMSDMEpDLL int `lbmsdm_msg_get_int64_name` (`lbmsdm_msg_t` \*Message, const char \*Name, int64\_t \*Value)

- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_uint64\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, uint64\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_float\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, float \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_double\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, double \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_decimal\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, lbmsdm\_decimal\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_timestamp\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, struct timeval \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_message\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, lbmsdm\_msg\_t \*\*Value)
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_get\\_boolean](#) (lbmsdm\_iter\_t \*Iterator, uint8\_t \*Value)  
*Fetch a field value from the field referenced by an iterator.*
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_get\\_int8](#) (lbmsdm\_iter\_t \*Iterator, int8\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_get\\_uint8](#) (lbmsdm\_iter\_t \*Iterator, uint8\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_get\\_int16](#) (lbmsdm\_iter\_t \*Iterator, int16\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_get\\_uint16](#) (lbmsdm\_iter\_t \*Iterator, uint16\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_get\\_int32](#) (lbmsdm\_iter\_t \*Iterator, int32\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_get\\_uint32](#) (lbmsdm\_iter\_t \*Iterator, uint32\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_get\\_int64](#) (lbmsdm\_iter\_t \*Iterator, int64\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_get\\_uint64](#) (lbmsdm\_iter\_t \*Iterator, uint64\_t \*Value)

- LBMSDMEExpDLL int [lbmsdm\\_iter\\_get\\_float](#) (lbmsdm\_iter\_t \*Iterator, float \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_get\\_double](#) (lbmsdm\_iter\_t \*Iterator, double \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_get\\_decimal](#) (lbmsdm\_iter\_t \*Iterator, lbmsdm\_decimal\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_get\\_timestamp](#) (lbmsdm\_iter\_t \*Iterator, struct timeval \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_get\\_message](#) (lbmsdm\_iter\_t \*Iterator, lbmsdm\_msg\_t \*\*Value)
- LBMSDMEExpDLL 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.*

- LBMSDMEExpDLL 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.*

- LBMSDMEExpDLL 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.*

- LBMSDMEExpDLL 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.*

- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_int8\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, size\_t Element, int8\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_uint8\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, size\_t Element, uint8\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_int16\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, size\_t Element, int16\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_uint16\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, size\_t Element, uint16\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_int32\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, size\_t Element, int32\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_uint32\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, size\_t Element, uint32\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_int64\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, size\_t Element, int64\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_uint64\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, size\_t Element, uint64\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_float\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, size\_t Element, float \*Value)

- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_double\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, size\_t Element, double \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_decimal\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, size\_t Element, lbmsdm\_decimal\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_timestamp\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, size\_t Element, struct timeval \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_message\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, size\_t Element, lbmsdm\_msg\_t \*\*Value)
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_int8\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, int8\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_uint8\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, uint8\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_int16\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, int16\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_uint16\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, uint16\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_int32\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, int32\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_uint32\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, uint32\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_int64\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, int64\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_uint64\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, uint64\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_float\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, float \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_get\\_double\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, double \*Value)

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

- LBMSDMEpDLL int `lbmsdm_msg_set_int64_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, int64\_t Value)
- LBMSDMEpDLL int `lbmsdm_msg_set_uint64_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, uint64\_t Value)
- LBMSDMEpDLL int `lbmsdm_msg_set_float_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, float Value)
- LBMSDMEpDLL int `lbmsdm_msg_set_double_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, double Value)
- LBMSDMEpDLL int `lbmsdm_msg_set_decimal_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, const `lbmsdm_decimal_t` \*Value)
- LBMSDMEpDLL int `lbmsdm_msg_set_timestamp_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, const struct timeval \*Value)
- LBMSDMEpDLL int `lbmsdm_msg_set_message_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, const `lbmsdm_msg_t` \*Value)
- LBMSDMEpDLL int `lbmsdm_msg_set_string_idx` (`lbmsdm_msg_t` \*Message, size\_t Index, const char \*Value)
- LBMSDMEpDLL 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.*

- LBMSDMEpDLL 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.*

- LBMSDMEpDLL 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.*

- LBMSDMEpDLL int `lbmsdm_msg_set_int8_name` (`lbmsdm_msg_t` \*Message, const char \*Name, int8\_t Value)
- LBMSDMEpDLL int `lbmsdm_msg_set_uint8_name` (`lbmsdm_msg_t` \*Message, const char \*Name, uint8\_t Value)
- LBMSDMEpDLL int `lbmsdm_msg_set_int16_name` (`lbmsdm_msg_t` \*Message, const char \*Name, int16\_t Value)
- LBMSDMEpDLL int `lbmsdm_msg_set_uint16_name` (`lbmsdm_msg_t` \*Message, const char \*Name, uint16\_t Value)
- LBMSDMEpDLL int `lbmsdm_msg_set_int32_name` (`lbmsdm_msg_t` \*Message, const char \*Name, int32\_t Value)
- LBMSDMEpDLL int `lbmsdm_msg_set_uint32_name` (`lbmsdm_msg_t` \*Message, const char \*Name, uint32\_t Value)
- LBMSDMEpDLL int `lbmsdm_msg_set_int64_name` (`lbmsdm_msg_t` \*Message, const char \*Name, int64\_t Value)
- LBMSDMEpDLL int `lbmsdm_msg_set_uint64_name` (`lbmsdm_msg_t` \*Message, const char \*Name, uint64\_t Value)

- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_float\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, float Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_double\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, double Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_decimal\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, const lbmsdm\_decimal\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_timestamp\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, const struct timeval \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_message\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, const lbmsdm\_msg\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_string\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, const char \*Value)
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_set\\_boolean](#) (lbmsdm\_iter\_t \*Iterator, uint8\_t Value)  
*Set a field value in the field referenced by an iterator.*
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_set\\_int8](#) (lbmsdm\_iter\_t \*Iterator, int8\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_set\\_uint8](#) (lbmsdm\_iter\_t \*Iterator, uint8\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_set\\_int16](#) (lbmsdm\_iter\_t \*Iterator, int16\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_set\\_uint16](#) (lbmsdm\_iter\_t \*Iterator, uint16\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_set\\_int32](#) (lbmsdm\_iter\_t \*Iterator, int32\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_set\\_uint32](#) (lbmsdm\_iter\_t \*Iterator, uint32\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_set\\_int64](#) (lbmsdm\_iter\_t \*Iterator, int64\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_set\\_uint64](#) (lbmsdm\_iter\_t \*Iterator, uint64\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_set\\_float](#) (lbmsdm\_iter\_t \*Iterator, float Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_set\\_double](#) (lbmsdm\_iter\_t \*Iterator, double Value)

- LBMSDMEExpDLL int [lbmsdm\\_iter\\_set\\_decimal](#) (lbmsdm\_iter\_t \*Iterator, const lbmsdm\_decimal\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_set\\_timestamp](#) (lbmsdm\_iter\_t \*Iterator, const struct timeval \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_set\\_message](#) (lbmsdm\_iter\_t \*Iterator, const lbmsdm\_msg\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_iter\\_set\\_string](#) (lbmsdm\_iter\_t \*Iterator, const char \*Value)
- LBMSDMEExpDLL 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.*

- LBMSDMEExpDLL 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.*

- LBMSDMEExpDLL 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.*

- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_int8\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_uint8\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_int16\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_uint16\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_int32\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_uint32\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_int64\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_uint64\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_float\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_double\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_decimal\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_timestamp\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)

- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_message\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_string\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_unicode\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_blob\\_array\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index)
- LBMSDMEExpDLL 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.*

- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_int8\\_array\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_uint8\\_array\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_int16\\_array\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_uint16\\_array\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_int32\\_array\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_uint32\\_array\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_int64\\_array\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_uint64\\_array\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_float\\_array\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_double\\_array\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_decimal\\_array\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_timestamp\\_array\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_message\\_array\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_string\\_array\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_unicode\\_array\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_blob\\_array\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name)

- LBMSDMEpDLL int `lbmsdm_iter_set_boolean_array` (`lbmsdm_iter_t *Iterator`)

*Set a field in a message by field name to an array field.*

- LBMSDMEpDLL int `lbmsdm_iter_set_int8_array` (`lbmsdm_iter_t *Iterator`)
- LBMSDMEpDLL int `lbmsdm_iter_set_uint8_array` (`lbmsdm_iter_t *Iterator`)
- LBMSDMEpDLL int `lbmsdm_iter_set_int16_array` (`lbmsdm_iter_t *Iterator`)
- LBMSDMEpDLL int `lbmsdm_iter_set_uint16_array` (`lbmsdm_iter_t *Iterator`)
- LBMSDMEpDLL int `lbmsdm_iter_set_int32_array` (`lbmsdm_iter_t *Iterator`)
- LBMSDMEpDLL int `lbmsdm_iter_set_uint32_array` (`lbmsdm_iter_t *Iterator`)
- LBMSDMEpDLL int `lbmsdm_iter_set_int64_array` (`lbmsdm_iter_t *Iterator`)
- LBMSDMEpDLL int `lbmsdm_iter_set_uint64_array` (`lbmsdm_iter_t *Iterator`)
- LBMSDMEpDLL int `lbmsdm_iter_set_float_array` (`lbmsdm_iter_t *Iterator`)
- LBMSDMEpDLL int `lbmsdm_iter_set_double_array` (`lbmsdm_iter_t *Iterator`)
- LBMSDMEpDLL int `lbmsdm_iter_set_decimal_array` (`lbmsdm_iter_t *Iterator`)
- LBMSDMEpDLL int `lbmsdm_iter_set_timestamp_array` (`lbmsdm_iter_t *Iterator`)
- LBMSDMEpDLL int `lbmsdm_iter_set_message_array` (`lbmsdm_iter_t *Iterator`)
- LBMSDMEpDLL int `lbmsdm_iter_set_string_array` (`lbmsdm_iter_t *Iterator`)
- LBMSDMEpDLL int `lbmsdm_iter_set_unicode_array` (`lbmsdm_iter_t *Iterator`)
- LBMSDMEpDLL int `lbmsdm_iter_set_blob_array` (`lbmsdm_iter_t *Iterator`)
- LBMSDMEpDLL 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.*

- LBMSDMEpDLL int `lbmsdm_msg_set_int8_elem_idx` (`lbmsdm_msg_t *Message`, `size_t Index`, `size_t Element`, `int8_t Value`)
- LBMSDMEpDLL int `lbmsdm_msg_set_uint8_elem_idx` (`lbmsdm_msg_t *Message`, `size_t Index`, `size_t Element`, `uint8_t Value`)
- LBMSDMEpDLL int `lbmsdm_msg_set_int16_elem_idx` (`lbmsdm_msg_t *Message`, `size_t Index`, `size_t Element`, `int16_t Value`)
- LBMSDMEpDLL int `lbmsdm_msg_set_uint16_elem_idx` (`lbmsdm_msg_t *Message`, `size_t Index`, `size_t Element`, `uint16_t Value`)
- LBMSDMEpDLL int `lbmsdm_msg_set_int32_elem_idx` (`lbmsdm_msg_t *Message`, `size_t Index`, `size_t Element`, `int32_t Value`)
- LBMSDMEpDLL int `lbmsdm_msg_set_uint32_elem_idx` (`lbmsdm_msg_t *Message`, `size_t Index`, `size_t Element`, `uint32_t Value`)
- LBMSDMEpDLL int `lbmsdm_msg_set_int64_elem_idx` (`lbmsdm_msg_t *Message`, `size_t Index`, `size_t Element`, `int64_t Value`)

- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_uint64\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, size\_t Element, uint64\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_float\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, size\_t Element, float Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_double\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, size\_t Element, double Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_decimal\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, size\_t Element, const lbmsdm\_decimal\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_timestamp\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, size\_t Element, const struct timeval \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_message\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, size\_t Element, const lbmsdm\_msg\_t \*Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_string\\_elem\\_idx](#) (lbmsdm\_msg\_t \*Message, size\_t Index, size\_t Element, const char \*Value)
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL 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.*
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_int8\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, int8\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_uint8\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, uint8\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_int16\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, int16\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_uint16\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, uint16\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_int32\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, int32\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_uint32\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, uint32\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_int64\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, int64\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_uint64\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, uint64\_t Value)
- LBMSDMEExpDLL int [lbmsdm\\_msg\\_set\\_float\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, float Value)

- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_double\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, double Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_decimal\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, const lbmsdm\_decimal\_t \*Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_timestamp\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, const struct timeval \*Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_message\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, const lbmsdm\_msg\_t \*Value)
- LBMSDMEpDLL int [lbmsdm\\_msg\\_set\\_string\\_elem\\_name](#) (lbmsdm\_msg\_t \*Message, const char \*Name, size\_t Element, const char \*Value)
- LBMSDMEpDLL 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.*

- LBMSDMEpDLL 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.*

- LBMSDMEpDLL 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.*

- LBMSDMEpDLL int [lbmsdm\\_iter\\_set\\_int8\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, int8\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_iter\\_set\\_uint8\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, uint8\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_iter\\_set\\_int16\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, int16\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_iter\\_set\\_uint16\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, uint16\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_iter\\_set\\_int32\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, int32\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_iter\\_set\\_uint32\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, uint32\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_iter\\_set\\_int64\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, int64\_t Value)
- LBMSDMEpDLL int [lbmsdm\\_iter\\_set\\_uint64\\_elem](#) (lbmsdm\_iter\_t \*Iterator, size\_t Element, uint64\_t Value)
- LBMSDMEpDLL 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\_6\_7\_1/29West/lbm/src/sdm/lbm/lbmsdm.h#1

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	<a href="#">LBMSDM_TYPE_-BOOLEAN</a>	uint8_t
8-bit signed integer	<a href="#">LBMSDM_TYPE_-INT8</a>	int8_t
8-bit unsigned integer	<a href="#">LBMSDM_TYPE_-UINT8</a>	uint8_t
16-bit signed integer	<a href="#">LBMSDM_TYPE_-INT16</a>	int16_t
16-bit unsigned integer	<a href="#">LBMSDM_TYPE_-UINT16</a>	uint16_t
32-bit signed integer	<a href="#">LBMSDM_TYPE_-INT32</a>	int32_t
32-bit unsigned integer	<a href="#">LBMSDM_TYPE_-UINT32</a>	uint32_t
64-bit signed integer	<a href="#">LBMSDM_TYPE_-INT64</a>	int64_t
64-bit unsigned integer	<a href="#">LBMSDM_TYPE_-UINT64</a>	uint64_t
Single-precision floating point	<a href="#">LBMSDM_TYPE_-FLOAT</a>	float
Double-precision floating point	<a href="#">LBMSDM_TYPE_-DOUBLE</a>	double
String	<a href="#">LBMSDM_TYPE_-STRING</a>	char *
Scaled decimal	<a href="#">LBMSDM_TYPE_-DECIMAL</a>	<a href="#">lbmsdm_decimal_t</a>
Timestamp	<a href="#">LBMSDM_TYPE_-TIMESTAMP</a>	struct timeval
Nested message	<a href="#">LBMSDM_TYPE_-MESSAGE</a>	<a href="#">lbmsdm_msg_t</a> *
Binary large object (BLOB)	<a href="#">LBMSDM_TYPE_-BLOB</a>	void *
Unicode string	<a href="#">LBMSDM_TYPE_-UNICODE</a>	wchar_t *

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 * sdmmmsg;
rc = lbmsdm_msg_parse(&sdmmmsg, lbmmmsg->data, lbmmmsg->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_elemflen_name()`, `lbmsdm_msg_get_elemflen_idx()`, and `lbmsdm_iter_get_elemflen()` 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																			
boolean	Yes	No	No	No	No	No	No	Yes												
int8	Yes	No	No	No	No	No	No	Yes												
uint8	Yes	No	No	No	No	No	No	Yes												
int16	Yes	No	No	No	No	No	No	Yes												
uint16	Yes	No	No	No	No	No	No	Yes												
int32	Yes	No	No	No	No	No	No	Yes												
uint32	Yes	No	No	No	No	No	No	Yes												
int64	Yes	No	No	No	No	No	No	Yes												
uint64	Yes	No	No	No	No	No	No	Yes												
float	Yes	No	No	No	No	No	No	No												
double	Yes	No	No	No	No	No	No	No												
string	No	Yes	No	No	No	No	No	No												
unicode	No	Yes	No	No	No	No	No													
timestamp	No	Yes	No	No	No	No														
BLOB	No	Yes	No	No	No															
messages	No	Yes	No	No																
decimal	Yes	No	No	No	No	No	No	Yes												

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 ≥ 0	32	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. 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.
				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 LBMSDMEpDLL const char\* lbmsdm\_errmsg (void)

**Returns:**

Pointer to a static char array containing the error message.

### 8.6.4.2 LBMSDMEpDLL int lbmsdm\_errnum (void)

**Returns:**

Integer error number (see LBMSDM\_ERROR\_\*).

### 8.6.4.3 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMEpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMEExpDLL 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 LBMSDMEExpDLL size\_t lbmsdm\_msg\_get\_dataalen ([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 LBMSDMEExpDLL int lbmsdm\_msg\_get\_elemcnt\_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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMEExpDLL int lbmsdm\_msg\_get\_fldcnt ([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 LBMSDMEExpDLL 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 LBMSDMEExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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 LBMSDMExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL 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** LBMSDMEExpDLL int lbmsdm\_win32\_static\_init (void)**Returns:**

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

## 8.7 umeblocksrc.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 **srp`, `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.

Copyright (c) 2005-2014 Informatica Corporation Permission is granted to licensees to use or alter this software for any purpose, including commercial applications, according to the terms laid out in the Software License Agreement.

This source code example is provided by Informatica for educational and evaluation purposes only.

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.

### 8.7.2 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* Pointer 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 UMEBlock 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

- [lbmontrlbm.h](#)
- [lbmontrlbm.c](#)

## 9.3 Source code for lbmmontrlbm.h

```
/** \file lbmmontrlbm.h
    \brief Ultra Messaging (UM) Monitoring API
    \author David K. Ameiss - Informatica Corporation
    \version $Id: //UMprod/REL_6_7_1/29West/lbm/src/mon/lbm/lbmmontrlbm.h#1 $

    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/lbmmon.h>

#ifdef __cplusplus
extern "C" {
#endif /* __cplusplus */

/** \brief Return a pointer to the LBMMON_TRANSPORT_LBM module structure.

    \return Pointer to LBMMON_TRANSPORT_LBM.
*/
LBMEXPDLL const lbmmon_transport_func_t * lbmmon_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
    lbmmon_sctl_create().
*/
```

```

        \return Zero if successful, -1 otherwise.
*/
LBMExpDLL int lbmmon_transport_lbm_initsrc(void * * TransportClientData,
                                           const void * TransportOptions);

/*!
 \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.
*/
LBMExpDLL int lbmmon_transport_lbm_initrcv(void * * TransportClientData,
                                           const void * TransportOptions);

/*!
 \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.
*/
LBMExpDLL 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.
*/
LBMExpDLL 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.
*/
LBMExpDLL 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.
*/
LBMExpDLL 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 * lbmon_transport_lbm_errmsg(void);

#ifdef __cplusplus
}
#endif /* __cplusplus */

#endif
```

## 9.4 Source code for lbmmontrlbm.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.

*/

#ifdef __VOS__
#define _POSIX_C_SOURCE 200112L
#include <sys/time.h>
#endif

#include <stdio.h>
#include <time.h>
#include <string.h>
#ifdef _WIN32
#define strcasecmp stricmp
#define snprintf _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/lbmmon.h>
#include <lbm/lbmmontrlbm.h>
#include <lbm/lbmaux.h>

/*
  Package all of the needed function pointers for this module into a

```

```

        lbmmon_transport_func_t structure.
*/
static const lbmmon_transport_func_t LBMMON_TRANSPORT_LBM =
{
    lbmmon_transport_lbm_initsrc,
    lbmmon_transport_lbm_initrcv,
    lbmmon_transport_lbm_send,
    lbmmon_transport_lbm_receive,
    lbmmon_transport_lbm_src_finish,
    lbmmon_transport_lbm_rcv_finish,
    lbmmon_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;
} lbmmon_transport_lbm_src_t;

/*
    A queue of incoming statistics packets is maintained. This describes each
    entry in the queue.
*/
struct lbmmon_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 lbmmon_transport_lbm_rcv_node_t_stct * mNext;
};
typedef struct lbmmon_transport_lbm_rcv_node_t_stct lbmmon_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 */
#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 */
    lbmmon_transport_lbm_rcv_node_t * mHead;
    /* Tail of the message queue */
    lbmmon_transport_lbm_rcv_node_t * mTail;
} lbmmon_transport_lbm_rcv_t;

static void src_cleanup(lbmmon_transport_lbm_src_t * Data);
static void rcv_cleanup(lbmmon_transport_lbm_rcv_t * Data);
static int receive_callback(lbm_rcv_t * Receiver, lbm_msg_t * Message, void * ClientData);
static void lock_receiver(lbmmon_transport_lbm_rcv_t * Receiver);
static void unlock_receiver(lbmmon_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 lbmmon_transport_func_t *
lbmmon_transport_lbm_module(void)
{
    return (&LBMMON_TRANSPORT_LBM);
}

int
lbmmon_transport_lbm_initsrc(void * * TransportClientData, const void * TransportOptions)
{
    lbmmon_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, sizeof(key), value, sizeof(value)))
{
    if (strcasemp(key, "config") == 0)
    {
        strncpy(config_file, value, sizeof(config_file));
    }
    else if (strcasemp(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_CON
if (rc == LBM_FAILURE)
{
    snprintf(ErrorString,
             sizeof(ErrorString),
             "lbm_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 attrib
    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());
        src_cleanup(data);
        return (-1);
    }
}
/* Go back through the options, looking for any specific context 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 (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),

```

```

        "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);
}
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++;
}
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 (strcasemp(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_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)
{
    lbmmon_transport_lbm_rcv_t * rcv = (lbmmon_transport_lbm_rcv_t *) ClientData;
    lbmmon_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 = lbmmon_next_key_value_pair(ptr, key, sizeof(key), value, sizeof(value))) != NULL)
{
    if (strcasemp(key, "config") == 0)
    {
        strncpy(config_file, value, sizeof(config_file));
    }
    else if (strcasemp(key, "topic") == 0)
    {
        strncpy(topic, value, sizeof(topic));
    }
    else if (strcasemp(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 = 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 (scope_is_valid(scope) == -1)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),
                 "invalid option scope [%s]",
                 scope);
        rcv_cleanup(data);
        return (-1);
    }
    if (strcasemp(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++;
}

/* Resolver cache need to enabled for wildcard receiver to work */
if (wildcard_topic[0] != '\0')
{
    rc = lbm_context_attr_str_setopt(data->mContextAttributes, "resolver_cache", "1");
    if (rc == LBM_FAILURE)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),

```

```

        "error setting option [context %s %s], %s",
        "resolver_cache",
        "1",
        lbm_errmsg());
    rcv_cleanup(data);
    return (rc);
}
}
/* 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 = lbmmon_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, value);
if (rc == LBM_FAILURE)
{
    snprintf(ErrorString,
              sizeof(ErrorString),
              "invalid option [wildcard_receiver %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, entry->value);
    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 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 = 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),
                    "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);
    }
}
}

```

```

#ifdef _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->mWildcardReceiver,
                                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),
             "lbm_wildcard_rcv_create()/lbm_rcv_create() failed, %s",
             lbm_errmsg());
    unlock_receiver(data);
    rcv_cleanup(data);
    return (rc);
}

/* Pass back the lbmmon_transport_lbm_rcv_t created */
*TransportClientData = data;
unlock_receiver(data);
return (0);
}

int
lbmmon_transport_lbm_send(const char * Data, size_t Length, void * TransportClientData)
{
    lbmmon_transport_lbm_src_t * src;
    int rc;

    if ((Data == NULL) || (TransportClientData == NULL))
    {
        strncpy(ErrorString, "Invalid argument", sizeof(ErrorString));
        return (-1);
    }
}

```

```

    }
    src = (lbmmon_transport_lbm_src_t *) TransportClientData;
    rc = lbm_src_send(src->mSource, Data, Length, 0);
    if (rc == LBM_FAILURE)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),
                 "lbm_src_send() failed, %s",
                 lbm_errmsg());
    }
    return (rc);
}

int
lbmmon_transport_lbm_receive(char * Data, size_t * Length, unsigned int TimeoutMS, void * TransportClientData)
{
    lbmmon_transport_lbm_rcv_t * rcv = (lbmmon_transport_lbm_rcv_t *) TransportClientData;
    lbmmon_transport_lbm_rcv_node_t * node;
    int rc = 0;
    size_t length_remaining;
#ifdef _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
    {
        /* MSGDESC: The monitoring message received is larger than the maximum
         * MSGRES: This is a hard coded maximum. */
        lbm_logf(LBM_LOG_ERR, "Core-8034-1: [LBMMON] Dropping monitoring message
                *Length, node->mMessage->len);
        /* 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);
        rc = 1; /* Positive number prevents caller from logging message too */
    }
    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
    tv.tv_sec = TimeoutMS / MILLISECONDS_PER_SECOND;
    tv.tv_nsec = (TimeoutMS % MILLISECONDS_PER_SECOND) * NANOSECONDS_PER_MILLISECOND;
    nanosleep(&tv, 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
lbmmon_transport_lbm_src_finish(void * TransportClientData)
{
    lbmmon_transport_lbm_src_t * src;

    if (TransportClientData == NULL)
    {
        strncpy(ErrorString, "Invalid argument", sizeof(ErrorString));
        return (-1);
    }
    src = (lbmmon_transport_lbm_src_t *) TransportClientData;
    src_cleanup(src);
    return (0);
}

void
rcv_cleanup(lbmmon_transport_lbm_rcv_t * Data)
{
    lbmmon_transport_lbm_rcv_node_t * node;
    lbmmon_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);
#ifdef _WIN32
        DeleteCriticalSection(&(Data->mLock));
#else
        pthread_mutex_destroy(&(Data->mLock));
#endif
    }

    free(Data);
}

int
lbmmon_transport_lbm_rcv_finish(void * TransportClientData)
{
    lbmmon_transport_lbm_rcv_t * rcv;

```

```
    if (TransportClientData == NULL)
    {
        strncpy(ErrorString, "Invalid argument", sizeof(ErrorString));
        return (-1);
    }
    rcv = (lbmmon_transport_lbm_rcv_t *) TransportClientData;
    rcv_cleanup(rcv);
    return (0);
}

void
lock_receiver(lbmmon_transport_lbm_rcv_t * Receiver)
{
#ifdef _WIN32
    EnterCriticalSection(&(Receiver->mLock));
#else
    pthread_mutex_lock(&(Receiver->mLock));
#endif
}

void
unlock_receiver(lbmmon_transport_lbm_rcv_t * Receiver)
{
#ifdef _WIN32
    LeaveCriticalSection(&(Receiver->mLock));
#else
    pthread_mutex_unlock(&(Receiver->mLock));
#endif
}

const char *
lbmmon_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

- [lbmontrudp.h](#)
- [lbmontrudp.c](#)

## 9.6 Source code for lbmmontrudp.h

```

/** \file lbmmontrudp.h
    \brief Ultra Messaging (UM) Monitoring API
    \author David K. Ameiss - Informatica Corporation
    \version $Id: //UMprod/REL_6_7_1/29West/lbm/src/mon/lbm/lbmmontrudp.h#1 $

    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 <lbm/lbmmon.h>

#ifdef __cplusplus
extern "C" {
#endif /* __cplusplus */

/*! \brief Return a pointer to the LBMMON_TRANSPORT_UDP module structure.

    \return Pointer to LBMMON_TRANSPORT_UDP.
*/
LBMExpDLL const lbmmon_transport_func_t * lbmmon_transport_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
    lbmmon_sctl_create().

```

```

        \return Zero if successful, -1 otherwise.
*/
LBMEExpDLL int lbmmon_transport_udp_initsrc(void * * TransportClientData,
                                            const void * TransportClientData);

/*!
 \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 * 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_udp_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_udp_receive(char * Data,
                                            size_t * Length,
                                            unsigned int TimeoutMS,
                                            void * TransportClientData);

/*!
 \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.
*/
LBMLExpDLL const char * lbmmon_transport_udp_errmsg(void);

#ifdef __cplusplus
}
#endif /* __cplusplus */

#endif
```

## 9.7 Source code for lbmmontrudp.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.

 */

#ifdef __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 stricmp
#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/lbmmon.h>
    #include <lbm/lbmmontrudp.h>

    #ifdef _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
        lbmmon_transport_func_t structure.
    */
    static const lbmmon_transport_func_t LBMMON_TRANSPORT_UDP =
    {
        lbmmon_transport_udp_initsrc,
        lbmmon_transport_udp_initrcv,
        lbmmon_transport_udp_send,
        lbmmon_transport_udp_receive,
        lbmmon_transport_udp_src_finish,
        lbmmon_transport_udp_rcv_finish,
        lbmmon_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;
    } lbmmon_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(lbmmon_transport_udp_rcv_t * Receiver);
static void unlock_receiver(lbmmon_transport_udp_rcv_t * Receiver);
#ifdef _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];
#ifdef _WIN32
    sprintf(message, sizeof(message), "error %d", WSAGetLastError());
#else
    sprintf(message,
            sizeof(message),
            "error %d, %s",
            errno,
            strerror(errno));
#endif
    return (message);
}

const lbmmon_transport_func_t *
lbmmon_transport_udp_module(void)
{
    return (&LBMMON_TRANSPORT_UDP);
}

int
lbmmon_transport_udp_initsrc(void * * TransportClientData, const void * TransportOptions)
{
    lbmmon_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(mcgrou, 0, sizeof(mcgrou));
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(mcgrou, value, sizeof(mcgrou));
    }
    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());
#ifdef _WIN32
        closesocket(data->mSocket);
#else
        close(data->mSocket);
#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, sizeof(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, sizeof(ifc_addr));
            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)
        {
#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);
    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 lbmmon_transport_udp_src_t created */
    *TransportClientData = data;
    return (0);
}

int
lbmmon_transport_udp_initrcv(void ** TransportClientData, const void * TransportOptions)
{

```

```

lbmmon_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(lbmmon_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 = lbmmon_next_key_value_pair(ptr, key, sizeof(key), value, sizeof(value))) != NULL)
{
    if (strcascmp(key, "port") == 0)
    {
        strncpy(port, value, sizeof(port));
    }
    else if (strcascmp(key, "interface") == 0)
    {
        strncpy(interface, value, sizeof(interface));
    }
    else if (strcascmp(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->mMulticastM
        if (rc == LBMMON_SOCKET_ERROR)
        {
            snprintf (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;

#ifdef _WIN32
    InitializeCriticalSection (&(data->mLock));
#else
    pthread_mutex_init (&(data->mLock), NULL);
#endif

    /* Start the receive thread */
#ifdef _WIN32
    data->mThread = CreateThread (NULL, 0, receive_thread_proc, data, 0, NULL);
    if (data->mThread == NULL)
    {
        snprintf (ErrorString,
                sizeof (ErrorString),
                "CreateThread() failed, error %d",
                GetLastError ());
        closesocket (data->mSocket);
        free (data);
        return (LBMMONTRUDP_ERR_THREAD);
    }
#else
#ifdef __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);
    }
#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 lbmmon_transport_udp_rcv_t created */
    *TransportClientData = data;
    return (0);
}

#ifdef _WIN32
DWORD WINAPI
receive_thread_proc(void * Arg)
#else
void *
receive_thread_proc(void * Arg)
#endif
{
    lbmmon_transport_udp_rcv_t * rcv = (lbmmon_transport_udp_rcv_t *) Arg;
    unsigned char buffer[8192];
    struct timeval timeout;
    fd_set readfds;
    int rc;
    ssize_t bytes_read;
    lbmmon_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);
    }
#ifdef _WIN32
    return (0);
#else
    return (NULL);
#endif
}

int
lbmmon_transport_udp_send(const char * Data, size_t Length, void * TransportClientData)
{
    lbmmon_transport_udp_src_t * src;
    int rc;

    if ((Data == NULL) || (TransportClientData == NULL))
    {
        return (-1);
    }
    src = (lbmmon_transport_udp_src_t *) TransportClientData;
    rc = sendto(src->mSocket, Data, Length, 0, (struct sockaddr *) &(src->mPeer), sizeof(src->mPeer));
    if (rc == LBMMON_SOCKET_ERROR)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),
                 "sendto() failed, %s",
                 last_socket_error());
        return (LBMMONTRUDP_ERR_SEND);
    }
    return (0);
}

int
lbmmon_transport_udp_receive(char * Data, size_t * Length, unsigned int TimeoutMS, void * TransportClientData)
{
    lbmmon_transport_udp_rcv_t * rcv = (lbmmon_transport_udp_rcv_t *) TransportClientData;
    lbmmon_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
        {
            /* MSGDESC: The monitoring message received is larger than the maximum
             * MSGRES: This is a hard coded maximum. */
            lbm_logf(LBM_LOG_ERR, "Core-8034-2: [LBMMON] Dropping monitoring message
                *Length, node->mMessageLength);
            /* 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);
        }
    }

```

```

        rc = 1; /* Positive number prevents caller from logging message too */
    }
    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);
}

int
lbmmon_transport_udp_src_finish(void * TransportClientData)
{
    lbmmon_transport_udp_src_t * src;

    if (TransportClientData == NULL)
    {
        strncpy(ErrorString, "Invalid parameter", sizeof(ErrorString));
        return (-1);
    }
    src = (lbmmon_transport_udp_src_t *) TransportClientData;

#ifdef _WIN32
    closesocket(src->mSocket);
#else
    close(src->mSocket);
#endif
    /* Clean up our data */
    free(TransportClientData);
    return (0);
}

```

```

int
lbmmon_transport_udp_rcv_finish(void * TransportClientData)
{
    lbmmon_transport_udp_rcv_t * rcv = (lbmmon_transport_udp_rcv_t *) TransportClientData;
    lbmmon_transport_udp_rcv_node_t * node;
    lbmmon_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->
#ifdef _WIN32
        closesocket(rcv->mSocket);
#else
        close(rcv->mSocket);
#endif
#ifdef _WIN32
        DeleteCriticalSection(&(rcv->mLock));
#else
        pthread_mutex_destroy(&(rcv->mLock));
#endif

        free(TransportClientData);
        return (0);
    }

void
lock_receiver(lbmmon_transport_udp_rcv_t * Receiver)
{
#ifdef _WIN32
    EnterCriticalSection(&(Receiver->mLock));
#else
    pthread_mutex_lock(&(Receiver->mLock));
#endif
}

```

```
void
unlock_receiver(lbmmon_transport_udp_rcv_t * Receiver)
{
#ifdef _WIN32
    LeaveCriticalSection(&(Receiver->mLock));
#else
    pthread_mutex_unlock(&(Receiver->mLock));
#endif
}

const char *
lbmmon_transport_udp_errmsg(void)
{
    return (ErrorString);
}
```

## 9.8 LBMMON CSV format module

- [lbmmonfmtcsv.h](#)
- [lbmmonfmtcsv.c](#)

## 9.9 Source code for lbmmonfmtcsv.h

```

/** \file lbmmonfmtcsv.h
    \brief Ultra Messaging (UM) Monitoring API
    \author David K. Ameiss - Informatica Corporation
    \version $Id: //UMprod/REL_6_7_1/29West/lbm/src/mon/lbm/lbmmonfmtcsv.h#1 $

    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 LBMMONFMTCVS_H
#define LBMMONFMTCVS_H

#include <stdlib.h>
#ifdef _WIN32
    #include <winsock2.h>
#endif
#include <lbm/lbmmon.h>

#ifdef __cplusplus
extern "C" {
#endif /* __cplusplus */

/*! \brief Return a pointer to the LBMMON_FORMAT_CSV module structure.

    \return Pointer to LBMMON_FORMAT_CSV.
*/
LBMExpDLL const lbmmon_format_func_t * lbmmon_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

```

```

        lbmmon_sctl_create() or lbmmon_rctl_create().
    \return Zero if successful, -1 otherwise.
        If -1 is returned,
            the serialized data will not be sent.
*/
LBMExpDLL int lbmmon_format_csv_init(void * * FormatClientData,
                                     const void * FormatOptions);

/*!
    \brief Serialize an ::lbm_rcv_transport_stats_t structure to CSV.

    \param Destination A pointer to a buffer to receive the serialized format
        of the ::lbm_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 ::lbm_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.
*/
LBMExpDLL int lbmmon_rcv_format_csv_serialize(char * Destination,
                                              size_t * Size,
                                              unsigned short * ModuleID,
                                              const void * Statistics);

/*!
    \brief Serialize an ::lbm_src_transport_stats_t structure to CSV.

    \param Destination A pointer to a buffer to receive the serialized format
        of the ::lbm_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 ::lbm_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.
*/
LBMExpDLL int lbmmon_src_format_csv_serialize(char * Destination,
                                              size_t * Size,
                                              unsigned short * ModuleID,
                                              const void * Statistics);

```

```

size_t * Size,
unsigned short *
const lbm_src_tr
void * FormatCli

/*!
\brief Serialize an ::lbm_event_queue_stats_t structure to CSV.

\param Destination A pointer to a buffer to receive the serialized format
of the ::lbm_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 ::lbm_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_evq_format_csv_serialize(char * Destination,

size_t * Size,
unsigned short *
const lbm_event_
void * FormatCli

/*!
\brief Serialize an ::lbm_context_stats_t structure to CSV.

\param Destination A pointer to a buffer to receive the serialized format
of the ::lbm_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 ::lbm_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_t * Size,
unsigned short *
const lbm_contex

```

```
/*! \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 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.
*/
LBMEExpDLL int lbmmon_rcv_format_csv_deserialize(lbm_rcv_transport_stats_t * Statistics,

/*! \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 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.
*/
LBMEExpDLL int lbmmon_src_format_csv_deserialize(lbm_src_transport_stats_t * Statistics,

/*! \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 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.
*/
LBMEExpDLL int lbmmon_evq_format_csv_deserialize(lbm_event_queue_stats_t * Statistics,

```

```

const char
size_t Len
unsigned s
void * For

```

```

//*!
    \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.
*/
LBMEExpDLL int lbmmon_ctx_format_csv_deserialize(lbm_context_stats_t * Statistics,

```

```

const char
size_t Len
unsigned s
void * For

```

```

//*!
    \brief Serialize receiver topic statistics.

    \param Destination A pointer to a buffer to receive the serialized format
           of the lbm_context_stats_t statistics.
    \param Size A pointer to a \c 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.
    \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 Topic A NUL-terminated string containing the topic.
    \param SourceCount The number of sources in the \a Sources array.
    \param Sources An array of ::lbm_rcv_topic_stats_t structures containing
           the sources to which the receiver is joined.
    \param FormatClientData A pointer to format-specific client data as
           returned by the ::lbmmon_format_init_t function.
    \return Zero if successful, -1 otherwise.
           If -1 is returned,
           the serialized data will not be sent.
*/
LBMEExpDLL int lbmmon_rcv_topic_format_csv_serialize(char * Destination,

```

```

si
un
cc

```

```

/*! \brief Deserialize a buffer into an ::lbm_rcv_topic_stats_t structure.

\param Count A pointer to an integer containing the number of elements in the
        \a Statistics array. On exit, it will contain the actual number of elements par
\param Statistics An array of ::lbm_rcv_topic_stats_t elements into which is written th
        actual deserialized data.
\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 as
        returned by the ::lbmmon_format_init_t function.
\return Zero if successful, -2 if \a Count is not large enough for all elements, -1 oth
        If -2 is returned, \a Count will contain the number of elements required.
        If -1 is returned,
        the deserialized data will not be delivered to the application.
*/
LBMEExpDLL int lbmmon_rcv_topic_format_csv_deserialize(size_t * Count,

/*! \brief Serialize wildcard receiver statistics.

\param Destination A pointer to a buffer to receive the serialized format
        of the lbm_wildcard_rcv_stats_t statistics.
\param Size A pointer to a \c 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.
\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 lbm_wildcard_rcv_stats_t structure to
        be serialized.
\param FormatClientData A pointer to format-specific client data as
        returned by the ::lbmmon_format_init_t function.
\return Zero if successful, -1 otherwise.
        If -1 is returned,
        the serialized data will not be sent.
*/
LBMEExpDLL int lbmmon_wildcard_rcv_format_csv_serialize(char * Destination,

```

```
/*!    \brief Deserialize a buffer into an ::lbm_wildcard_rcv_stats_t structure.

    \param Statistics A pointer to an lbm_wildcard_rcv_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 as
           returned by the ::lbmmon_format_init_t function.
    \return Zero if successful, -1 otherwise.
           If -1 is returned,
           the deserialized data will not be delivered to the application.
*/
LBMEExpDLL int lbmmon_wildcard_rcv_format_csv_deserialize(lbm_wildcard_rcv_stats_t * Statistics,

/*!    \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.
*/
LBMEExpDLL int lbmmon_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.
*/
LBMEExpDLL const char * lbmmon_format_csv_errmsg(void);

#if defined(__cplusplus)
}
#endif /* __cplusplus */

#endif
```

## 9.10 Source code for lbmmonfmtcsv.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.

*/

#ifdef __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>
#include <ctype.h>
#ifdef _WIN32
#define strcasecmp stricmp
#define snprintf _snprintf
#else
#endif
#ifdef __TANDEM__
#include <strings.h>
#endif
#include <lbm/lbmmon.h>
#include <lbm/lbmmonfmtcsv.h>
static const lbmmon_format_func_t LBMMON_FORMAT_CSV =
{
    lbmmon_format_csv_init,
    lbmmon_rcv_format_csv_serialize,
    lbmmon_src_format_csv_serialize,
    lbmmon_rcv_format_csv_deserialize,
    lbmmon_src_format_csv_deserialize,
    lbmmon_format_csv_finish,
    lbmmon_format_csv_errmsg,
    lbmmon_evq_format_csv_serialize,

```

```

    lbmmon_evq_format_csv_deserialize,
    lbmmon_ctx_format_csv_serialize,
    lbmmon_ctx_format_csv_deserialize,
    lbmmon_rcv_topic_format_csv_serialize,
    lbmmon_rcv_topic_format_csv_deserialize,
    lbmmon_wildcard_rcv_format_csv_serialize,
    lbmmon_wildcard_rcv_format_csv_deserialize
};

typedef struct
{
    unsigned char mSeparator;
    size_t mBufferSize;
    char * mBuffer;
} lbmmon_format_csv_t;

static const char * next_csv_value(const char * String, char * Value, size_t Size, char Separator);

#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_5 5
#define LBMMON_FORMAT_CSV_VERSION_CURRENT LBMMON_FORMAT_CSV_VERSION_5
#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;
} lbmmon_csv_layout_t;

static void lbmmon_format_csv_convert_to_hex(char * Buffer, const lbm_uint8_t * Data, size_t Length)
{
    static char hextable[16] =
        { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f' };
    unsigned char c;
    char * buf = Buffer;
    size_t iidx = 0;
    size_t oidx = 0;

    while (iidx < Length)
    {
        c = Data[iidx];
        buf[oidx++] = hextable[((c >> 4) & 0x0f)];
        buf[oidx++] = hextable[(c & 0x0f)];
        iidx++;
    }
}

static void lbmmon_format_csv_convert_from_hex(char * Buffer, const lbm_uint8_t * Data, size_t Length)
{
    unsigned char c;
    char * buf = Buffer;

```

```
size_t iidx = 0;
size_t oidx = 0;
unsigned char n1;
unsigned char n2;
unsigned char b;

if ((Length % 2) != 0)
{
    memset((void *) Buffer, 0, (Length / 2));
    return;
}
while (iidx < Length)
{
    b = 0;
    c = Data[iidx++];
    if (isxdigit(c))
    {
        if (isdigit(c))
        {
            n1 = c - '0';
        }
        else if (isupper(c))
        {
            n1 = 0x0a + (c - 'A');
        }
        else
        {
            n1 = 0x0a + (c - 'a');
        }
    }
    else
    {
        n1 = 0;
    }
    c = Data[iidx++];
    if (isxdigit(c))
    {
        if (isdigit(c))
        {
            n2 = c - '0';
        }
        else if (isupper(c))
        {
            n2 = 0x0a + (c - 'A');
        }
        else
        {
            n2 = 0x0a + (c - 'a');
        }
    }
    else
    {
        n2 = 0;
    }
    b = ((n1 & 0x0f) << 4) | (n2 & 0x0f);
    buf[oidx++] = b;
}
```

```

}

static char ErrorString[1024];

const lbmmon_format_func_t * lbmmon_format_csv_module(void)
{
    return (&LBMMON_FORMAT_CSV);
}

int lbmmon_format_csv_init(void ** FormatClientData, const void * FormatOptions)
{
    char key[512];
    char value[512];
    const char * ptr = (const char *) FormatOptions;
    lbmmon_format_csv_t * data;

    memset(ErrorString, 0, sizeof(ErrorString));
    data = malloc(sizeof(lbmmon_format_csv_t));
    data->mSeparator = ',';
    data->mBufferSize = 1024;
    data->mBuffer = malloc(data->mBufferSize);

    while ((ptr = lbmmon_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_* values)
source (as a string)
actual statistics, depending on type...
for LBTRM:
    msgs_rcved
    bytes_rcved
    nak_pkts_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_twx
    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_pkts_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 lbmmon_rcv_format_csv_serialize(char * Destination, size_t * Size, unsigned short * Module)

```

```

const lbm_rcv_transport_stats_t +
{
    char work[1024];
    lbmmon_format_csv_t      * fmt;

    if ((Destination == NULL) || (Size == NULL) || (*Size == 0) || (Statistics == NULL) || (FormatClientData == NULL))
    {
        return (-1);
    }
    fmt = (lbmmon_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%c%lx",
                    fmt->mSeparator,
                    Statistics->transport.lbtrm.msgs_rcved,
                    fmt->mSeparator,
                    Statistics->transport.lbtrm.bytes_rcved,
                    fmt->mSeparator,
                    Statistics->transport.lbtrm.nak_pkts_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(D
    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.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(D
    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.unrecovered_twx,
        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(D

```

```

        return (-1);
    }
    strcat(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(ErrorString));
        return (-1);
    }
    strcat(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);
    }
    strcat(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.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_twx,
        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,
        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);
snprintf(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:

```

```

        snprintf(work,
                 sizeof(work),
                 "%c%c%c%c%c%c%c%c",
                 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_LBTSMX:
        snprintf(work,
                 sizeof(work),
                 "%c%c%c%c%c%c%c%c",
                 fmt->mSeparator,
                 Statistics->transport.lbtismx.msgs_rcved,
                 fmt->mSeparator,
                 Statistics->transport.lbtismx.bytes_rcved,
                 fmt->mSeparator,
                 Statistics->transport.lbtismx.lbm_msgs_rcved,
                 fmt->mSeparator,
                 Statistics->transport.lbtismx.lbm_msgs_no_topic_rcved,
                 fmt->mSeparator,
                 Statistics->transport.lbtismx.reserved1);
        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:
        snprintf(work,
                 sizeof(work),
                 "%c%c%c%c%c%c%c%c",
                 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
        rctrlr_data_msgs
        rctrlr_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 LBTIPC:
        num_clients
            msgs_sent
            bytes_sent
    for LBTRDMA:
        num_clients
            msgs_sent
            bytes_sent

```

```

*/
int lbmmon_src_format_csv_serialize(char * Destination, size_t * Size, unsigned short * ModuleID,
                                   const lbm_src_transport_stats_t *
{
    char work[1024];
    lbmmon_format_csv_t      * fmt;

    if ((Destination == NULL) || (Size == NULL) || (*Size == 0) || (Statistics == NULL) || (FormatClientData == NULL))
    {
        return (-1);
    }
    fmt = (lbmmon_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%c%c",
                    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%c%c%c%c%c%c%c%c%c%c",
                    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_pkts_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(Destination));
    return (-1);
}
strncat(Destination, work, *Size - strlen(Destination) - 1);
snprintf(work,
        sizeof(work),
        "%c%lx%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(Destination));
    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_pkts_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(Destination));
    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(ErrorString));
    return (-1);
}
strncat(Destination, work, *Size - strlen(Destination) - 1);
break;

case LBM_TRANSPORT_STAT_LBTSMX:
    snprintf(work,
        sizeof(work),
        "%c%lx%c%lx%c%lx",
        fmt->mSeparator,
        Statistics->transport.lbtismx.num_clients,
        fmt->mSeparator,
        Statistics->transport.lbtismx.msgs_sent,
        fmt->mSeparator,
        Statistics->transport.lbtismx.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_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

```



```

        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",
        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),
        "%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c",
        Statistics->source_events,
        fmt->mSeparator,

```

```

        Statistics->source_events_tot,
        fmt->mSeparator,
        Statistics->source_events_svc_min,
        fmt->mSeparator,
        Statistics->source_events_svc_mean,
        fmt->mSeparator,
        Statistics->source_events_svc_max,
        fmt->mSeparator,
        Statistics->unlock_events,
        fmt->mSeparator,
        Statistics->unlock_events_tot,
        fmt->mSeparator,
        Statistics->cancel_events,
        fmt->mSeparator,
        Statistics->cancel_events_tot,
        fmt->mSeparator,
        Statistics->cancel_events_svc_min,
        fmt->mSeparator,
        Statistics->cancel_events_svc_mean,
        fmt->mSeparator,
        Statistics->cancel_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),
        "%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx%c%lx",
        Statistics->context_source_events,
        fmt->mSeparator,
        Statistics->context_source_events_tot,
        fmt->mSeparator,
        Statistics->context_source_events_svc_min,
        fmt->mSeparator,
        Statistics->context_source_events_svc_mean,
        fmt->mSeparator,
        Statistics->context_source_events_svc_max,
        fmt->mSeparator,
        Statistics->events,
        fmt->mSeparator,
        Statistics->events_tot,
        fmt->mSeparator,
        Statistics->age_min,
        fmt->mSeparator,
        Statistics->age_mean,
        fmt->mSeparator,
        Statistics->age_max,
        fmt->mSeparator,
        Statistics->callback_events,
        fmt->mSeparator,
        Statistics->callback_events_tot,
        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
fragments_lost
fragments_unrecoverably_lost
rcv_cb_svc_time_min
rcv_cb_svc_time_max
rcv_cb_svc_time_mean
*/

int lbmmon_ctx_format_csv_serialize(char * Destination, size_t * Size, unsigned short * ModuleID,
                                   const lbm_context_statistics * Statistics)
{
    lbmmon_format_csv_t * fmt;
    char work[1024];

    if ((Destination == NULL) || (Size == NULL) || (*Size == 0) || (Statistics == NULL) ||
        (Statistics->callback_events_svc_max == 0))
    {
        return (-1);
    }
    fmt = (lbmmon_format_csv_t *) FormatClientData;

    *ModuleID = MAKE_MODULE_VERSION(LBMMON_FORMAT_CSV_VERSION_CURRENT);
    memset(work, 0, sizeof(work));
    memset(Destination, 0, 2048);
    snprintf(work,

```



```

        Statistics->rcv_cb_svc_time_mean);
    if (strlen(work) >= (*Size - strlen(Destination) - 1))
    {
        strncpy(ErrorString, "Destination too small for data", sizeof(ErrorString));
        return (-1);
    }
    strcat(Destination, work, *Size - strlen(Destination) - 1);
    *Size = strlen(Destination);
    return (0);
}

/*
Format of the CSV receiver topic statistics data is:
topic
source_count
For each source:
    source_string
    OTID
    topic_idx
*/

int lbmmon_rcv_topic_format_csv_serialize(char * Destination, size_t * Size, unsigned short * M
                                            lbm_ulong_t S
{
    lbmmon_format_csv_t * fmt;
    char work[1024];
    int idx;

    if ((Destination == NULL) || (Size == NULL) || (*Size == 0) || (FormatClientData == NU
    {
        return (-1);
    }
    fmt = (lbmmon_format_csv_t *) FormatClientData;

    *ModuleID = MAKE_MODULE_VERSION(LBMMON_FORMAT_CSV_VERSION_CURRENT);
    memset(work, 0, sizeof(work));
    memset(Destination, 0, 2048);
    snprintf(work, sizeof(work),
             "\"%s\" \"%c%lx\"",
             Topic,
             fmt->mSeparator,
             SourceCount);
    if (strlen(work) >= (*Size - strlen(Destination) - 1))
    {
        strncpy(ErrorString, "Destination too small for data", sizeof(ErrorString));
        return (-1);
    }
    strcat(Destination, work, *Size - strlen(Destination) - 1);
    if (SourceCount > 0)
    {
        for (idx = 0; idx < SourceCount; ++idx)
        {
            size_t offset;
            memset(work, 0, sizeof(work));
            snprintf(work, sizeof(work),
                    "%c\"%s\" \"%c\"",
                    fmt->mSeparator,

```

```

                Sources[idx].source,
                fmt->mSeparator);
        offset = strlen(work);
        lbmmon_format_csv_convert_to_hex(work + offset, Sources[idx].otid, LBM_OTID_BLOCK);
        offset = strlen(work);
        snprintf(work + offset, sizeof(work) - offset,
                "%c%x",
                fmt->mSeparator,
                Sources[idx].topic_idx);
        if (strlen(work) >= (*Size - strlen(Destination) - 1))
        {
                strncpy(ErrorString, "Destination too small for data", sizeof(ErrorString));
                return (-1);
        }
        strcat(Destination, work, *Size - strlen(Destination) - 1);
    }
    *Size = strlen(Destination);
    return (0);
}

/*
Format of the CSV wildcard receiver statistics data is:
pattern
type
*/

int lbmmon_wildcard_rcv_format_csv_serialize(char * Destination, size_t * Size, unsigned short * ModuleID,
                                            const lbm_wildcard_receiver_statistics_t * Statistics)
{
    lbmmon_format_csv_t * fmt;
    char work[1024];

    if ((Destination == NULL) || (Size == NULL) || (*Size == 0) || (Statistics == NULL) || (FormatClientData == NULL))
    {
        return (-1);
    }
    fmt = (lbmmon_format_csv_t *) FormatClientData;

    *ModuleID = MAKE_MODULE_VERSION(LBMMON_FORMAT_CSV_VERSION_CURRENT);
    memset(work, 0, sizeof(work));
    memset(Destination, 0, 2048);
    snprintf(work, sizeof(work),
            "%s%c",
            Statistics->pattern,
            fmt->mSeparator,
            Statistics->type);
    if (strlen(work) >= (*Size - strlen(Destination) - 1))
    {
        strncpy(ErrorString, "Destination too small for data", sizeof(ErrorString));
        return (-1);
    }
    strcat(Destination, work, *Size - strlen(Destination) - 1);
    *Size = strlen(Destination);
    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 separate
/* parsing loops for each of n message types, a single function could parse the string and put
/* the values into the structure (given the appropriate pointers). Access to the statistics
/* 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 the
/* 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
#define csv_rcv_tcp_stat_offset_v5 csv_rcv_tcp_stat_offset_v4

```

```

static const lbmmon_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) },
    { csv_rcv_tcp_stat_offset_v5, sizeof(csv_rcv_tcp_stat_offset_v5)/sizeof(csv_rcv_tcp_stat_offset_v5) },
};

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_twx),
    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_twx),
    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
#define csv_rcv_lbtrm_stat_offset_v5 csv_rcv_lbtrm_stat_offset_v4
static const lbmmon_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_off
    { csv_rcv_lbtrm_stat_offset_v2, sizeof(csv_rcv_lbtrm_stat_offset_v2)/sizeof(csv_rcv_lbtrm_stat_off
    { csv_rcv_lbtrm_stat_offset_v3, sizeof(csv_rcv_lbtrm_stat_offset_v3)/sizeof(csv_rcv_lbtrm_stat_off
    { csv_rcv_lbtrm_stat_offset_v4, sizeof(csv_rcv_lbtrm_stat_offset_v4)/sizeof(csv_rcv_lbtrm_stat_off
    { csv_rcv_lbtrm_stat_offset_v5, sizeof(csv_rcv_lbtrm_stat_offset_v5)/sizeof(csv_rcv_lbtrm_stat_off
};

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
#define csv_rcv_lbtru_stat_offset_v5 csv_rcv_lbtru_stat_offset_v4
static const lbmmon_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) },
    { csv_rcv_lbtru_stat_offset_v5, sizeof(csv_rcv_lbtru_stat_offset_v5)/sizeof(csv_rcv_lbtru_stat_offset_v5) }
};

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
#define csv_rcv_lbtipc_stat_offset_v5 csv_rcv_lbtipc_stat_offset_v4
static const lbmmon_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) },
    { csv_rcv_lbtipc_stat_offset_v5, sizeof(csv_rcv_lbtipc_stat_offset_v5)/sizeof(csv_rcv_lbtipc_stat_offset_v5) }
};

static const size_t csv_rcv_lbtismx_stat_offset_v2[] =
{
    offsetof(lbm_rcv_transport_stats_lbtismx_t, msgs_rcved),
    offsetof(lbm_rcv_transport_stats_lbtismx_t, bytes_rcved),
    offsetof(lbm_rcv_transport_stats_lbtismx_t, lbm_msgs_rcved),
    offsetof(lbm_rcv_transport_stats_lbtismx_t, lbm_msgs_no_topic_rcved),

```

```

        offsetof(lbm_rcv_transport_stats_lbtsmx_t, reserved1)
};
#define csv_rcv_lbtsmx_stat_offset_v3 csv_rcv_lbtsmx_stat_offset_v2
#define csv_rcv_lbtsmx_stat_offset_v4 csv_rcv_lbtsmx_stat_offset_v3
#define csv_rcv_lbtsmx_stat_offset_v5 csv_rcv_lbtsmx_stat_offset_v4
static const lbmmon_csv_layout_t csv_rcv_lbtsmx_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT+1] =
{
    { NULL, 0 },
    { NULL, 0 },
    { csv_rcv_lbtsmx_stat_offset_v2, sizeof(csv_rcv_lbtsmx_stat_offset_v2)/sizeof(csv_rcv_lbtsmx_stat_offset_v2) },
    { csv_rcv_lbtsmx_stat_offset_v3, sizeof(csv_rcv_lbtsmx_stat_offset_v3)/sizeof(csv_rcv_lbtsmx_stat_offset_v3) },
    { csv_rcv_lbtsmx_stat_offset_v4, sizeof(csv_rcv_lbtsmx_stat_offset_v4)/sizeof(csv_rcv_lbtsmx_stat_offset_v4) },
    { csv_rcv_lbtsmx_stat_offset_v5, sizeof(csv_rcv_lbtsmx_stat_offset_v5)/sizeof(csv_rcv_lbtsmx_stat_offset_v5) }
};

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)
};
#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
#define csv_rcv_lbtrdma_stat_offset_v5 csv_rcv_lbtrdma_stat_offset_v4
static const lbmmon_csv_layout_t csv_rcv_lbtrdma_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT+1] =
{
    { 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) },
    { csv_rcv_lbtrdma_stat_offset_v5, sizeof(csv_rcv_lbtrdma_stat_offset_v5)/sizeof(csv_rcv_lbtrdma_stat_offset_v5) }
};

int lbmmon_rcv_format_csv_deserialize(lbm_rcv_transport_stats_t * Statistics, const char * Source, size_t Length,
                                     unsigned short ModuleID, void * ClientData)
{
    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;
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];
        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_tcp));
    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) + idx)) =
            strtoul(value, NULL, 10);
    }
    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];
        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));

    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 stats,
             * older releases of lbm may have 26 fields and newer releases may
             * See bug 5002 for more information.
             * For version 3, we will not consider it an error if there are only 26 fields.
             */
            if(modver == MODULE_VERSION(3) && idx == 26) {
                return 0;
            }

            strncpy(ErrorString, "Data contains too few fields", sizeof(ErrorString));
            return (-1);
        }
        *((lbm_ulong_t *) ((unsigned char *)&(Statistics->transport.lbtrm) + stat_count * idx)) =
    }
    break;

case LBM_TRANSPORT_STAT_LBTRU:
    if (modver >= LBMMON_FORMAT_CSV_VERSION_CURRENT)
    {
        stat_layout = csv_rcv_lbtru_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT];
        stat_count = csv_rcv_lbtru_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].count;
    }
    else
    {
        stat_layout = csv_rcv_lbtru_stat_layout[modver].layout;
        stat_count = csv_rcv_lbtru_stat_layout[modver].count;
    }
    memset((void *) &(Statistics->transport.lbtru), 0, sizeof(lbm_rcv_transport_stats));
    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.lbtru) + stat_count * idx)) =
    }
}

```

```

        break;

    case LBM_TRANSPORT_STAT_LBTIPC:
        if (modver >= LBMMON_FORMAT_CSV_VERSION_CURRENT)
        {
            stat_layout = csv_rcv_lbtipc_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT];
            stat_count = csv_rcv_lbtipc_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].count;
        }
        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_transport_stat_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.lbtipc[idx]))) = strtoul(value, &ptr, 10);
        }
        break;

    case LBM_TRANSPORT_STAT_LBTSMX:
        if (modver >= LBMMON_FORMAT_CSV_VERSION_CURRENT)
        {
            stat_layout = csv_rcv_lbtsmx_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT];
            stat_count = csv_rcv_lbtsmx_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].count;
        }
        else
        {
            stat_layout = csv_rcv_lbtsmx_stat_layout[modver].layout;
            stat_count = csv_rcv_lbtsmx_stat_layout[modver].count;
        }
        memset((void *) &(Statistics->transport.lbtsmx), 0, sizeof(lbm_rcv_transport_stat_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.lbtsmx[idx]))) = strtoul(value, &ptr, 10);
        }
        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].count;
        }
        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_stats));
        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_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
#define csv_src_tcp_stat_offset_v5 csv_src_tcp_stat_offset_v4
static const lbmmon_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) },
    { csv_src_tcp_stat_offset_v5, sizeof(csv_src_tcp_stat_offset_v5)/sizeof(csv_src_tcp_stat_offset_v5) }
};

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
#define csv_src_lbtrm_stat_offset_v5 csv_src_lbtrm_stat_offset_v4
static const lbmmon_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) },
    { csv_src_lbtrm_stat_offset_v5, sizeof(csv_src_lbtrm_stat_offset_v5)/sizeof(csv_src_lbtrm_stat_offset_v5) }
};

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

```

```

#define csv_src_lbtru_stat_offset_v5 csv_src_lbtru_stat_offset_v4
static const lbmmon_csv_layout_t csv_src_lbtru_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT+1] =
{
    { NULL, 0 },
    { csv_src_lbtru_stat_offset_v1, sizeof(csv_src_lbtru_stat_offset_v1)/sizeof(csv_src_lbtru_stat_off
    { csv_src_lbtru_stat_offset_v2, sizeof(csv_src_lbtru_stat_offset_v2)/sizeof(csv_src_lbtru_stat_off
    { csv_src_lbtru_stat_offset_v3, sizeof(csv_src_lbtru_stat_offset_v3)/sizeof(csv_src_lbtru_stat_off
    { csv_src_lbtru_stat_offset_v4, sizeof(csv_src_lbtru_stat_offset_v4)/sizeof(csv_src_lbtru_stat_off
    { csv_src_lbtru_stat_offset_v5, sizeof(csv_src_lbtru_stat_offset_v5)/sizeof(csv_src_lbtru_stat_off
};

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
#define csv_src_lbtipc_stat_offset_v5 csv_src_lbtipc_stat_offset_v4
static const lbmmon_csv_layout_t csv_src_lbtipc_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT+1] =
{
    { NULL, 0 },
    { NULL, 0 },
    { csv_src_lbtipc_stat_offset_v2, sizeof(csv_src_lbtipc_stat_offset_v2)/sizeof(csv_src_lbtipc_stat
    { csv_src_lbtipc_stat_offset_v3, sizeof(csv_src_lbtipc_stat_offset_v3)/sizeof(csv_src_lbtipc_stat
    { csv_src_lbtipc_stat_offset_v4, sizeof(csv_src_lbtipc_stat_offset_v4)/sizeof(csv_src_lbtipc_stat
    { csv_src_lbtipc_stat_offset_v5, sizeof(csv_src_lbtipc_stat_offset_v5)/sizeof(csv_src_lbtipc_stat
};

static size_t csv_src_lbtsmx_stat_offset_v2[] =
{
    offsetof(lbm_src_transport_stats_lbtsmx_t, num_clients),
    offsetof(lbm_src_transport_stats_lbtsmx_t, msgs_sent),
    offsetof(lbm_src_transport_stats_lbtsmx_t, bytes_sent)
};
#define csv_src_lbtsmx_stat_offset_v3 csv_src_lbtsmx_stat_offset_v2
#define csv_src_lbtsmx_stat_offset_v4 csv_src_lbtsmx_stat_offset_v3
#define csv_src_lbtsmx_stat_offset_v5 csv_src_lbtsmx_stat_offset_v4
static const lbmmon_csv_layout_t csv_src_lbtsmx_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT+1] =
{
    { NULL, 0 },
    { NULL, 0 },
    { csv_src_lbtsmx_stat_offset_v2, sizeof(csv_src_lbtsmx_stat_offset_v2)/sizeof(csv_src_lbtsmx_stat
    { csv_src_lbtsmx_stat_offset_v3, sizeof(csv_src_lbtsmx_stat_offset_v3)/sizeof(csv_src_lbtsmx_stat
    { csv_src_lbtsmx_stat_offset_v4, sizeof(csv_src_lbtsmx_stat_offset_v4)/sizeof(csv_src_lbtsmx_stat
    { csv_src_lbtsmx_stat_offset_v5, sizeof(csv_src_lbtsmx_stat_offset_v5)/sizeof(csv_src_lbtsmx_stat
};

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

```

```

#define csv_src_lbtrdma_stat_offset_v5 csv_src_lbtrdma_stat_offset_v4
static const lbmmon_csv_layout_t csv_src_lbtrdma_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT]
{
    { NULL, 0 },
    { NULL, 0 },
    { csv_src_lbtrdma_stat_offset_v2, sizeof(csv_src_lbtrdma_stat_offset_v2)/sizeof(csv_src_lbtrdma_stat_offset_v2) },
    { csv_src_lbtrdma_stat_offset_v3, sizeof(csv_src_lbtrdma_stat_offset_v3)/sizeof(csv_src_lbtrdma_stat_offset_v3) },
    { csv_src_lbtrdma_stat_offset_v4, sizeof(csv_src_lbtrdma_stat_offset_v4)/sizeof(csv_src_lbtrdma_stat_offset_v4) },
    { csv_src_lbtrdma_stat_offset_v5, sizeof(csv_src_lbtrdma_stat_offset_v5)/sizeof(csv_src_lbtrdma_stat_offset_v5) }
};

int lbmmon_src_format_csv_deserialize(lbm_src_transport_stats_t * Statistics, const char * Source, unsigned short ModuleID)
{
    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) || (ModuleID == 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].l
            stat_count = csv_src_tcp_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].co
        }
        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(ErrorS
                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_src_lbtrm_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT]
            stat_count = csv_src_lbtrm_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].
        }
        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(ErrorS
                return (-1);
            }
            *((lbm_ulong_t *) ((unsigned char *)&(Statistics->transport.lbtrm)) + stat
        }
        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];
        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_stat_lbtru));
    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.lbtru[idx])));
    }
    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_stat_lbtipc));
    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[idx])));
    }
    break;

case LBM_TRANSPORT_STAT_LBTSMX:
    if (modver >= LBMMON_FORMAT_CSV_VERSION_CURRENT)
    {
        stat_layout = csv_src_lbtsmx_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT];
        stat_count = csv_src_lbtsmx_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT].count;
    }
    else
    {

```

```

        stat_layout = csv_src_lbtsmx_stat_layout[modver].layout;
        stat_count = csv_src_lbtsmx_stat_layout[modver].count;
    }
    memset((void *) &(Statistics->transport.lbtsmx), 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);
        }
        *((lbm_ulong_t *)(((unsigned char *)&(Statistics->transport.lbtsmx)) + stat_count * idx)) = strtoul(ptr, &ptr, 10);
    }
    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_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.lbtrdma)) + stat_count * idx)) = strtoul(ptr, &ptr, 10);
    }
    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, wrvc_msgs),
offsetof(lbm_event_queue_stats_t, wrvc_msgs_tot),
offsetof(lbm_event_queue_stats_t, wrvc_msgs_svc_min),
offsetof(lbm_event_queue_stats_t, wrvc_msgs_svc_mean),
offsetof(lbm_event_queue_stats_t, wrvc_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, wrvc_msgs),
offsetof(lbm_event_queue_stats_t, wrvc_msgs_tot),
offsetof(lbm_event_queue_stats_t, wrvc_msgs_svc_min),
offsetof(lbm_event_queue_stats_t, wrvc_msgs_svc_mean),
offsetof(lbm_event_queue_stats_t, wrvc_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
#define csv_evq_stat_offset_v5 csv_evq_stat_offset_v4
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]) },

```

```

        { csv_evq_stat_offset_v5, sizeof(csv_evq_stat_offset_v5)/sizeof(csv_evq_stat_offset_v5)
};

int lbmmon_evq_format_csv_deserialize(lbm_event_queue_stats_t * Statistics, const char * Source,
                                     unsigned short ModuleID)
{
    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) ||
        (ModuleID == 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;
    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_value(value);
    }
    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 size_t csv_ctx_stat_offset_v5[] =
{
    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),
offsetof(lbm_context_stats_t, fragments_lost),
offsetof(lbm_context_stats_t, fragments_unrecoverably_lost),
offsetof(lbm_context_stats_t, rcv_cb_svc_time_min),
offsetof(lbm_context_stats_t, rcv_cb_svc_time_max),
offsetof(lbm_context_stats_t, rcv_cb_svc_time_mean)
};
static const lbmmon_csv_layout_t csv_ctx_stat_layout [LBMMON_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) },
    { csv_ctx_stat_offset_v3, sizeof(csv_ctx_stat_offset_v3)/sizeof(csv_ctx_stat_offset_v3) },
    { csv_ctx_stat_offset_v4, sizeof(csv_ctx_stat_offset_v4)/sizeof(csv_ctx_stat_offset_v4) },
    { csv_ctx_stat_offset_v5, sizeof(csv_ctx_stat_offset_v5)/sizeof(csv_ctx_stat_offset_v5) }
};

int lbmmon_ctx_format_csv_deserialize(lbm_context_stats_t * Statistics, const char * Source, signed short ModuleID)
{
    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;
    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_value(value,
    }
    return (0);
}

int lbmmon_rcv_topic_format_csv_deserialize(size_t * Count, lbm_rcv_topic_stats_t * Statistics, const char * Source,
                                           size_t Length, unsigned char modid, unsigned char modver)
{
    const char * ptr;
    char value[1024];
    lbmmon_format_csv_t * fmt;
    size_t idx;
    unsigned char modid;
    unsigned char modver;
    char topic[LBM_MSG_MAX_TOPIC_LEN + 1];
    size_t src_count;

    if ((Count == NULL) || (*Count == 0) || (Statistics == NULL) || (Source == NULL) || (*Source == '\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 (modver != LBMMON_FORMAT_CSV_VERSION_5)
    {
        strncpy(ErrorString, "Unknown version", 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, "Data contains too few fields", sizeof(ErrorString));
        return (-1);
    }
    strncpy(topic, value, sizeof(topic));
    ptr = next_csv_value(ptr, value, sizeof(value), fmt->mSeparator);
    if (ptr == NULL)
    {
        strncpy(ErrorString, "Data contains too few fields", sizeof(ErrorString));
        return (-1);
    }
    src_count = (size_t) convert_value(value);
    if (*Count < src_count)
    {
        /* Not enough entries. */
        *Count = src_count;
        return (-2);
    }
    if (src_count == 0)
    {
        memset((void *) &(Statistics[0]), 0, sizeof(lbm_rcv_topic_stats_t));
        strncpy(Statistics[0].topic, topic, sizeof(Statistics[idx].topic));
        src_count = 1;
    }

```

```

    }
    else
    {
        for (idx = 0; idx < src_count; ++idx)
        {
            memset((void *) &(Statistics[idx]), 0, sizeof(lbm_rcv_topic_stats_t));
            strncpy(Statistics[idx].topic, topic, sizeof(Statistics[idx].topic));
            Statistics[idx].flags |= LBM_RCV_TOPIC_STATS_FLAG_SRC_VALID;
            ptr = next_csv_value(ptr, value, sizeof(value), fmt->mSeparator);
            if (ptr == NULL)
            {
                strncpy(ErrorString, "Data contains too few fields", sizeof(ErrorString));
                return (-1);
            }
            strncpy(Statistics[idx].source, (void *) value, sizeof(Statistics[idx].source));
            ptr = next_csv_value(ptr, value, sizeof(value), fmt->mSeparator);
            if (ptr == NULL)
            {
                strncpy(ErrorString, "Data contains too few fields", sizeof(ErrorString));
                return (-1);
            }
            lbmmon_format_csv_convert_from_hex((char *) &(Statistics[idx].otid), (lbm_uint8_t *) value);
            ptr = next_csv_value(ptr, value, sizeof(value), fmt->mSeparator);
            if (ptr == NULL)
            {
                strncpy(ErrorString, "Data contains too few fields", sizeof(ErrorString));
                return (-1);
            }
            Statistics[idx].topic_idx = (lbm_uint32_t) convert_value(value);
        }
    }
    *Count = src_count;
    return (0);
}

/* Note: these are currently not used. */
static size_t csv_wrcv_stat_offset_v5[] =
{
    offsetof(lbm_wildcard_rcv_stats_t, pattern),
    offsetof(lbm_wildcard_rcv_stats_t, type)
};

static const lbmmon_csv_layout_t csv_wrcv_stat_layout[LBMMON_FORMAT_CSV_VERSION_CURRENT+1] =
{
    { NULL, 0 },
    { csv_wrcv_stat_offset_v5, sizeof(csv_wrcv_stat_offset_v5)/sizeof(csv_wrcv_stat_offset_v5[0]) }
};

int lbmmon_wildcard_rcv_format_csv_deserialize(lbm_wildcard_rcv_stats_t * Statistics, const char * Source,
                                              size_t Length,
                                              {
            const char * ptr;
            char value[1024];
            lbmmon_format_csv_t * fmt;

```

```

unsigned char modid;
unsigned char modver;

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 (modver != LBMMON_FORMAT_CSV_VERSION_5)
{
    strncpy(ErrorString, "Unknown version", 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;
memset((void *) Statistics, 0, sizeof(lbm_wildcard_rcv_stats_t));
ptr = next_csv_value(ptr, value, sizeof(value), fmt->mSeparator);
if (ptr == NULL)
{
    strncpy(ErrorString, "Data contains too few fields", sizeof(ErrorString));
    return (-1);
}
strncpy(Statistics->pattern, value, sizeof(Statistics->pattern));
ptr = next_csv_value(ptr, value, sizeof(value), fmt->mSeparator);
if (ptr == NULL)
{
    strncpy(ErrorString, "Data contains too few fields", sizeof(ErrorString));
    return (-1);
}
Statistics->type = (lbm_uint8_t) convert_value(value);
return (0);
}

int lbmmon_format_csv_finish(void * FormatClientData)
{

```

```
    if (FormatClientData != NULL)
    {
        lbmmon_format_csv_t * data = (lbmmon_format_csv_t *) FormatClientData;
        if (data->mBuffer != NULL)
        {
            free(data->mBuffer);
            data->mBuffer = NULL;
        }
        free(data);
    }
    return (0);
}

const char * lbmmon_format_csv_errmsg(void)
{
    return (ErrorString);
}
```

## 9.11 LBMMON LBMSNMP transport module

- [lbmontrlbsnmp.h](#)
- [lbmontrlbsnmp.c](#)

## 9.12 Source code for lbmmontrlbmsnmp.h

```

/** \file lbmmontrlbmsnmp.h
    \brief Ultra Messaging (UM) Monitoring API
    \author David K. Ameiss - Informatica Corporation
    \version $Id: //UMprod/REL_6_7_1/29West/lbm/src/mon/lbm/lbmmontrlbmsnmp.h#1 $

    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/lbmmon.h>

#ifdef __cplusplus
extern "C" {
#endif /* __cplusplus */

/*! \brief Return a pointer to the LBMMON_TRANSPORT_LBMSNMP module structure.

    \return Pointer to LBMMON_TRANSPORT_LBMSNMP.
*/
LBMEExpDLL const lbmmon_transport_func_t * lbmmon_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
    lbmmon_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
uns:
void *

/!*
    \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 * lbmmon_transport_lbsnmp_errmsg(void);

#ifdef __cplusplus
}
#endif /* __cplusplus */

#endif
```

## 9.13 Source code for lbmmontrlbmsnmp.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.

*/

#ifdef __VOS__
#define _POSIX_C_SOURCE 200112L
#include <sys/time.h>
#endif

#include <stdio.h>
#include <time.h>
#include <string.h>
#ifdef _WIN32
#define strcasecmp stricmp
#define snprintf _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/lbmmon.h>
#include <lbm/lbmmontrlbmsnmp.h>
#include <lbm/lbmaux.h>

/*
  Package all of the needed function pointers for this module into a

```

```

        lbmmon_transport_func_t structure.
*/
static const lbmmon_transport_func_t LBMMON_TRANSPORT_LBMSNMP =
{
    lbmmon_transport_lbmsnmp_initsrc,
    lbmmon_transport_lbmsnmp_initrcv,
    lbmmon_transport_lbmsnmp_send,
    lbmmon_transport_lbmsnmp_receive,
    lbmmon_transport_lbmsnmp_src_finish,
    lbmmon_transport_lbmsnmp_rcv_finish,
    lbmmon_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;
} lbmmon_transport_lbmsnmp_src_t;

/*
    A queue of incoming statistics packets is maintained. This describes each
    entry in the queue.
*/
struct lbmmon_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 lbmmon_transport_lbmsnmp_rcv_node_t_stct * mNext;
};
typedef struct lbmmon_transport_lbmsnmp_rcv_node_t_stct lbmmon_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 */
    lbmmon_transport_lbmsnmp_rcv_node_t * mHead;
    /* Tail of the message queue */
    lbmmon_transport_lbmsnmp_rcv_node_t * mTail;
} lbmmon_transport_lbmsnmp_rcv_t;

static void src_cleanup(lbmmon_transport_lbmsnmp_src_t * Data);
static void rcv_cleanup(lbmmon_transport_lbmsnmp_rcv_t * Data);
static int receive_callback(lbm_rcv_t * Receiver, lbm_msg_t * Message, void * ClientData);
static void lock_receiver(lbmmon_transport_lbmsnmp_rcv_t * Receiver);
static void unlock_receiver(lbmmon_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_LBTRM_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 lbmmon_transport_func_t *
lbmmon_transport_lbmsnmp_module(void)
{
    return (&LBMMON_TRANSPORT_LBMSNMP);
}

int
lbmmon_transport_lbmsnmp_initsrc(void ** TransportClientData, const void * TransportOptions)
{
    lbmmon_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(lbmmon_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 = lbmmon_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)
    {

```

```

        snprintf(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 = lbmmon_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 (scope_is_valid(scope) == -1)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),
                 "invalid option scope [%s]",
                 scope);
        src_cleanup(data);
        return (-1);
    }
    if (strcasemp(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 = &SourceContextOptionFixed[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),
                 "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)))
{
    if (sscanf(key, "[%a-zA-Z_]|[%a-zA-Z_]", scope, option) != 2)
    {
        continue;
    }
    if (strcasemp(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 = lbmmon_next_key_value_pair(ptr, key, sizeof(key), value, sizeof(value)))
{
    if (strcasemp(key, "config") == 0)
    {
        strncpy(config_file, value, sizeof(config_file));
    }
    else if (strcasemp(key, "topic") == 0)
    {
        strncpy(topic, value, sizeof(topic));
    }
    else if (strcasemp(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_CON
if (rc == LBM_FAILURE)
{
    snprintf(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
    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 = 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 (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->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++;
    }

    /* Resolver cache need to enabled for wildcard receiver to work */
    if (wildcard_topic[0] != '\0')
    {
        rc = lbm_context_attr_str_setopt(data->mContextAttributes, "resolver_cache", "1");
        if (rc == LBM_FAILURE)
        {
            snprintf(ErrorString,
                     sizeof(ErrorString),
                     "error setting option [context %s %s], %s",
                     "resolver_cache",
                     "1",
                     lbm_errmsg());
            rcv_cleanup(data);
            return (rc);
        }
    }

    /* 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 = lbmmon_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, entry->op
    if (rc == LBM_FAILURE)
    {
        snprintf(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)
{
    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)))
{
    if (sscanf(key, "[%a-zA-Z_]|%[a-zA-Z_]", scope, option) != 2)
    {
        continue;
    }
    if (strcasemp(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),
                    "lbm_rcv_topic_lookup() failed, %s",
                    lbm_errmsg());
            rcv_cleanup(data);
            return (rc);
        }
    }

#ifdef _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),
                 "lbm_wildcard_rcv_create()/lbm_rcv_create() failed, %s",
                 lbm_errmsg());
        unlock_receiver(data);
        rcv_cleanup(data);
        return (rc);
    }

    /* Pass back the lbmmon_transport_lbmsnmp_rcv_t created */
    *TransportClientData = data;
    unlock_receiver(data);
    return (0);
}

int
lbmmon_transport_lbmsnmp_send(const char * Data, size_t Length, void * TransportClientData)
{
    lbmmon_transport_lbmsnmp_src_t * src;
    int rc;

    if ((Data == NULL) || (TransportClientData == NULL))
    {
        strncpy(ErrorString, "Invalid argument", sizeof(ErrorString));
        return (-1);
    }
    src = (lbmmon_transport_lbmsnmp_src_t *) TransportClientData;
    rc = lbm_src_send(src->mSource, Data, Length, 0);
    if (rc == LBM_FAILURE)
    {
        snprintf(ErrorString,
                 sizeof(ErrorString),
                 "lbm_src_send() failed, %s",
                 lbm_errmsg());
    }
    return (rc);
}

int
lbmmon_transport_lbmsnmp_receive(char * Data, size_t * Length, unsigned int TimeoutMS, void * T
{
    lbmmon_transport_lbmsnmp_rcv_t * rcv = (lbmmon_transport_lbmsnmp_rcv_t *) TransportCli
    lbmmon_transport_lbmsnmp_rcv_node_t * node;
    int rc = 0;
    size_t length_remaining;
#ifdef _WIN32
#elif defined(__TANDEM)
    unsigned int sleep_sec;
    unsigned int sleep_usec;
#else
    struct timespec tv;
#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
    {
        /* MSGDESC: The monitoring message received is larger than the maximum allowed size
        * MSGRES: This is a hard coded maximum. */
        lbm_logf(LBM_LOG_ERR, "Core-8034-3: [LBMMON] Dropping monitoring message that is larger than
        *Length, node->mMessage->len);
        /* 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);
        rc = 1; /* Positive number prevents caller from logging message too */
    }
    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(lbmon_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
lbmon_transport_lbmsnmp_src_finish(void * TransportClientData)
{
    lbmon_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);
#ifdef _WIN32
    DeleteCriticalSection(&(Data->mLock));
#else
    pthread_mutex_destroy(&(Data->mLock));
#endif
}

free(Data);
}

int
lbmmon_transport_lbmsnmp_rcv_finish(void * TransportClientData)
{
    lbmmon_transport_lbmsnmp_rcv_t * rcv;

    if (TransportClientData == NULL)
    {
        strncpy(ErrorString, "Invalid argument", sizeof(ErrorString));
        return (-1);
    }
    rcv = (lbmmon_transport_lbmsnmp_rcv_t *) TransportClientData;
    rcv_cleanup(rcv);
    return (0);
}

void
lock_receiver(lbmmon_transport_lbmsnmp_rcv_t * Receiver)
{
#ifdef _WIN32
    EnterCriticalSection(&(Receiver->mLock));
#else
    pthread_mutex_lock(&(Receiver->mLock));
#endif
}

void
unlock_receiver(lbmmon_transport_lbmsnmp_rcv_t * Receiver)
{
#ifdef _WIN32
    LeaveCriticalSection(&(Receiver->mLock));
#else
    pthread_mutex_unlock(&(Receiver->mLock));
}

```

```
#endif
}

const char *
lbmmon_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 [lbm\\_msg\\_gateway\\_info\\_t\\_stct](#)

Global [LBMMON\\_ATTR\\_CONTEXTID](#) Use [LBMMON\\_ATTR\\_OBJECTID](#) instead.

Global [lbmmon\\_attr\\_get\\_contextid](#) Use [lbmmon\\_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, 149
- age\_max
  - lbm\_event\_queue\_stats\_t\_stct, 135
- age\_mean
  - lbm\_event\_queue\_stats\_t\_stct, 135
- age\_min
  - lbm\_event\_queue\_stats\_t\_stct, 136
- apphdr\_chain
  - lbm\_src\_send\_ex\_info\_t\_stct, 231
- application\_set\_index
  - lbm\_src\_event\_umq\_ulb\_message\_-  
info\_ex\_t\_stct, 225
  - lbm\_src\_event\_umq\_ulb\_receiver\_-  
info\_ex\_t\_stct, 227
  - lbm\_umq\_ulb\_receiver\_type\_-  
entry\_t\_stct, 280
- appname
  - lbm\_umm\_info\_t\_stct, 268
- appsets
  - lbm\_umq\_queue\_topic\_t\_stct, 277
- assignment\_id
  - lbm\_msg\_umq\_registration\_-  
complete\_ex\_t\_stct, 171
  - lbm\_src\_event\_umq\_ulb\_message\_-  
info\_ex\_t\_stct, 225

- lbm\_src\_event\_umq\_ulb\_receiver\_info\_ex\_t\_stct, 227
- async\_opfunc
  - lbm\_src\_send\_ex\_info\_t\_stct, 231
- bits
  - lbm\_ipv4\_address\_mask\_t\_stct, 149
- bytes
  - lbm\_flight\_size\_inflight\_t\_stct, 146
- bytes\_buffered
  - lbm\_src\_transport\_stats\_daemon\_t\_stct, 233
  - lbm\_src\_transport\_stats\_tcp\_t\_stct, 247
- bytes\_rcved
  - lbm\_rcv\_transport\_stats\_daemon\_t\_stct, 176
  - lbm\_rcv\_transport\_stats\_lbtipc\_t\_stct, 177
  - lbm\_rcv\_transport\_stats\_lbtrdma\_t\_stct, 179
  - lbm\_rcv\_transport\_stats\_lbtrm\_t\_stct, 181
  - lbm\_rcv\_transport\_stats\_lbtru\_t\_stct, 188
  - lbm\_rcv\_transport\_stats\_lbtsmx\_t\_stct, 194
  - lbm\_rcv\_transport\_stats\_tcp\_t\_stct, 199
- bytes\_sent
  - lbm\_src\_transport\_stats\_lbtipc\_t\_stct, 234
  - lbm\_src\_transport\_stats\_lbtrdma\_t\_stct, 235
  - lbm\_src\_transport\_stats\_lbtrm\_t\_stct, 236
  - lbm\_src\_transport\_stats\_lbtru\_t\_stct, 240
  - lbm\_src\_transport\_stats\_lbtsmx\_t\_stct, 243
- callback\_events
  - lbm\_event\_queue\_stats\_t\_stct, 136
- callback\_events\_svc\_max
  - lbm\_event\_queue\_stats\_t\_stct, 136
- callback\_events\_svc\_mean
  - lbm\_event\_queue\_stats\_t\_stct, 136
- callback\_events\_svc\_min
  - lbm\_event\_queue\_stats\_t\_stct, 136
- callback\_events\_tot
  - lbm\_event\_queue\_stats\_t\_stct, 136
- cancel\_events
  - lbm\_event\_queue\_stats\_t\_stct, 137
- cancel\_events\_svc\_max
  - lbm\_event\_queue\_stats\_t\_stct, 137
- cancel\_events\_svc\_mean
  - lbm\_event\_queue\_stats\_t\_stct, 137
- cancel\_events\_svc\_min
  - lbm\_event\_queue\_stats\_t\_stct, 137
- cancel\_events\_tot
  - lbm\_event\_queue\_stats\_t\_stct, 137
- cbfunc
  - lbmmon\_ctx\_statistics\_func\_t\_stct, 287
  - lbmmon\_evq\_statistics\_func\_t\_stct, 288
  - lbmmon\_rcv\_statistics\_func\_t\_stct, 294
  - lbmmon\_rcv\_topic\_statistics\_func\_t\_stct, 295
  - lbmmon\_src\_statistics\_func\_t\_stct, 296
  - lbmmon\_wildcard\_rcv\_statistics\_func\_t\_stct, 299
- cbproc
  - lbm\_delete\_cb\_info\_t\_stct, 132
  - lbm\_event\_queue\_cancel\_cb\_info\_t\_stct, 133
- cert\_file
  - lbm\_umm\_info\_t\_stct, 268
- cert\_file\_password
  - lbm\_umm\_info\_t\_stct, 268
- channel\_info
  - lbm\_msg\_t\_stct, 156
  - lbm\_src\_send\_ex\_info\_t\_stct, 231
- channel\_number
  - lbm\_msg\_channel\_info\_t\_stct, 151
- clientd
  - lbm\_async\_operation\_func\_t, 115
  - lbm\_delete\_cb\_info\_t\_stct, 132
  - lbm\_event\_queue\_cancel\_cb\_info\_t\_stct, 133

- lbm\_umq\_queue\_msg\_status\_t, 274
- context\_source\_events
  - lbm\_event\_queue\_stats\_t\_stct, 137
- context\_source\_events\_svc\_max
  - lbm\_event\_queue\_stats\_t\_stct, 137
- context\_source\_events\_svc\_mean
  - lbm\_event\_queue\_stats\_t\_stct, 138
- context\_source\_events\_svc\_min
  - lbm\_event\_queue\_stats\_t\_stct, 138
- context\_source\_events\_tot
  - lbm\_event\_queue\_stats\_t\_stct, 138
- copied\_state
  - lbm\_msg\_t\_stct, 156
- d
  - lbm\_umq\_ulb\_application\_set\_attr\_t\_stct, 278
  - lbm\_umq\_ulb\_receiver\_type\_attr\_t\_stct, 279
- daemon
  - lbm\_rcv\_transport\_stats\_t\_stct, 197
  - lbm\_src\_transport\_stats\_t\_stct, 245
- data
  - lbm\_apphdr\_chain\_elem\_t\_stct, 113
  - lbm\_msg\_t\_stct, 156
- data\_msgs
  - lbm\_event\_queue\_stats\_t\_stct, 138
- data\_msgs\_svc\_max
  - lbm\_event\_queue\_stats\_t\_stct, 138
- data\_msgs\_svc\_mean
  - lbm\_event\_queue\_stats\_t\_stct, 139
- data\_msgs\_svc\_min
  - lbm\_event\_queue\_stats\_t\_stct, 139
- data\_msgs\_tot
  - lbm\_event\_queue\_stats\_t\_stct, 139
- dest\_port
  - lbm\_transport\_source\_info\_t\_stct, 251
- destination\_port
  - lbm\_ucast\_resolver\_entry\_t\_stct, 253
- dgrams\_dropped\_hdr
  - lbm\_rcv\_transport\_stats\_lbtrm\_t\_stct, 181
  - lbm\_rcv\_transport\_stats\_lbtru\_t\_stct, 188
- 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, 189
- 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
- domain\_id
  - lbm\_ume\_store\_entry\_t\_stct, 264
- duplicate\_data
  - lbm\_rcv\_transport\_stats\_lbtrm\_t\_stct, 182
  - lbm\_rcv\_transport\_stats\_lbtru\_t\_stct, 189
- event\_queue
  - lbm\_event\_queue\_cancel\_cb\_info\_t\_stct, 133
- events
  - lbm\_event\_queue\_stats\_t\_stct, 139
- events\_tot
  - lbm\_event\_queue\_stats\_t\_stct, 139
- evq
  - lbm\_async\_operation\_func\_t, 115
- exp
  - lbmpdm\_decimal\_t, 300
  - lbmsdm\_decimal\_t\_stct, 305
- field\_type
  - lbmpdm\_field\_value\_stct\_t, 302

- first\_sequence\_number
  - lbm\_src\_event\_sequence\_number\_info\_t\_stct, 209
  - lbm\_src\_event\_umq\_stability\_ack\_info\_ex\_t\_stct, 223
  - lbm\_src\_event\_umq\_ulb\_message\_info\_ex\_t\_stct, 225
- fixed\_str\_len
  - lbmpdm\_field\_info\_attr\_stct\_t, 301
- 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, 151
  - lbm\_msg\_t\_stct, 156
  - lbm\_msg\_ume\_deregistration\_ex\_t\_stct, 160
  - lbm\_msg\_ume\_registration\_complete\_ex\_t\_stct, 162
  - lbm\_msg\_ume\_registration\_ex\_t\_stct, 163
  - lbm\_msg\_umq\_deregistration\_complete\_ex\_t\_stct, 166
  - lbm\_msg\_umq\_index\_assigned\_ex\_t\_stct, 167
  - lbm\_msg\_umq\_index\_assignment\_eligibility\_start\_complete\_ex\_t\_stct, 168
  - lbm\_msg\_umq\_index\_assignment\_eligibility\_stop\_complete\_ex\_t\_stct, 169
  - lbm\_msg\_umq\_index\_released\_ex\_t\_stct, 170
  - lbm\_msg\_umq\_registration\_complete\_ex\_t\_stct, 171
  - lbm\_rcv\_topic\_stats\_t\_stct, 174
  - lbm\_src\_event\_sequence\_number\_info\_t\_stct, 209
  - lbm\_src\_event\_ume\_ack\_ex\_info\_t\_stct, 211
  - lbm\_src\_event\_ume\_deregistration\_ex\_t\_stct, 214
  - lbm\_src\_event\_ume\_registration\_complete\_ex\_t\_stct, 216
  - lbm\_src\_event\_ume\_registration\_ex\_t\_stct, 217
  - lbm\_src\_event\_umq\_message\_id\_info\_t\_stct, 220
  - lbm\_src\_event\_umq\_registration\_complete\_ex\_t\_stct, 222
  - lbm\_src\_event\_umq\_stability\_ack\_info\_ex\_t\_stct, 223
  - lbm\_src\_event\_umq\_ulb\_message\_info\_ex\_t\_stct, 226
  - lbm\_src\_event\_umq\_ulb\_receiver\_info\_ex\_t\_stct, 227
  - lbm\_src\_event\_wakeup\_t\_stct, 229
  - lbm\_src\_send\_ex\_info\_t\_stct, 232
  - lbm\_ume\_rcv\_recovery\_info\_ex\_func\_info\_t\_stct, 256
  - lbm\_ume\_rcv\_regid\_ex\_func\_info\_t\_stct, 259
  - lbm\_umm\_info\_t\_stct, 268
  - lbm\_umq\_index\_info\_t\_stct, 270
  - lbm\_umq\_msg\_total\_lifetime\_info\_t\_stct, 271
  - lbm\_umq\_queue\_msg\_status\_t, 274
  - lbm\_umq\_queue\_topic\_status\_t, 276
- fragments\_lost
  - lbm\_context\_stats\_t\_stct, 126
- fragments\_unrecoverably\_lost
  - lbm\_context\_stats\_t\_stct, 126
- 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 referenced 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
  - lbm\_ume\_store\_entry\_t\_stct, [264](#)
  - lbm\_ume\_store\_name\_entry\_t\_stct, [267](#)
- group\_size
  - lbm\_ume\_store\_group\_entry\_t\_stct, [266](#)
- handle
  - lbm\_async\_operation\_info\_t, [117](#)
- hashfunc
  - lbm\_str\_hash\_func\_ex\_t\_stct, [248](#)
- hf\_sequence\_number
  - lbm\_msg\_t\_stct, [156](#)
- hf\_sqn
  - lbm\_src\_send\_ex\_info\_t\_stct, [232](#)
- high\_sequence\_number
  - lbm\_ume\_rcv\_recovery\_info\_ex\_func\_info\_t\_stct, [256](#)
- id
  - lbm\_umq\_ulb\_receiver\_type\_attr\_t\_stct, [279](#)
  - lbm\_umq\_ulb\_receiver\_type\_entry\_t\_stct, [280](#)
- iface
  - lbm\_ucast\_resolver\_entry\_t\_stct, [253](#)
- index
  - lbm\_ume\_store\_group\_entry\_t\_stct, [266](#)
  - lbm\_umq\_index\_info\_t\_stct, [270](#)
  - lbm\_umq\_ulb\_application\_set\_attr\_t\_stct, [278](#)
- index\_len
  - lbm\_umq\_index\_info\_t\_stct, [270](#)
- info
  - lbm\_async\_operation\_info\_t, [117](#)

- io\_events
  - lbm\_event\_queue\_stats\_t\_stct, 139
- io\_events\_svc\_max
  - lbm\_event\_queue\_stats\_t\_stct, 139
- io\_events\_svc\_mean
  - lbm\_event\_queue\_stats\_t\_stct, 140
- io\_events\_svc\_min
  - lbm\_event\_queue\_stats\_t\_stct, 140
- io\_events\_tot
  - lbm\_event\_queue\_stats\_t\_stct, 140
- iov\_base
  - lbm\_iovec\_t\_stct, 148
- iov\_len
  - lbm\_iovec\_t\_stct, 148
- ip\_address
  - lbm\_ume\_store\_entry\_t\_stct, 264
- is\_array
  - lbmpdm\_field\_value\_stct\_t, 302
- is\_fixed
  - lbmpdm\_field\_value\_stct\_t, 302
- last\_sequence\_number
  - lbm\_src\_event\_sequence\_number\_info\_t\_stct, 209
  - lbm\_src\_event\_umq\_stability\_ack\_info\_ex\_t\_stct, 223
  - lbm\_src\_event\_umq\_ulb\_message\_info\_ex\_t\_stct, 226
- lbm.h, 309
  - lbm\_apphdr\_chain\_append\_elem, 434
  - lbm\_apphdr\_chain\_create, 435
  - lbm\_apphdr\_chain\_delete, 435
  - lbm\_apphdr\_chain\_iter\_create, 435
  - lbm\_apphdr\_chain\_iter\_create\_from\_msg, 435
  - lbm\_apphdr\_chain\_iter\_current, 436
  - lbm\_apphdr\_chain\_iter\_delete, 436
  - lbm\_apphdr\_chain\_iter\_done, 436
  - lbm\_apphdr\_chain\_iter\_first, 437
  - lbm\_apphdr\_chain\_iter\_next, 437
  - LBM\_ASYNC\_OP\_INFO\_FLAG\_FIRST, 377
  - LBM\_ASYNC\_OP\_INFO\_FLAG\_INLINE, 377
  - LBM\_ASYNC\_OP\_INFO\_FLAG\_LAST, 377
  - LBM\_ASYNC\_OP\_INFO\_FLAG\_ONLY, 377
  - LBM\_ASYNC\_OP\_INVALID\_HANDLE, 377
  - LBM\_ASYNC\_OP\_STATUS\_CANCELED, 377
  - LBM\_ASYNC\_OP\_STATUS\_COMPLETE, 377
  - LBM\_ASYNC\_OP\_STATUS\_ERROR, 378
  - LBM\_ASYNC\_OP\_STATUS\_IN\_PROGRESS, 378
  - LBM\_ASYNC\_OP\_TYPE\_CTX\_UMQ\_QUEUE\_TOPIC\_LIST, 378
  - LBM\_ASYNC\_OP\_TYPE\_RCV\_UMQ\_QUEUE\_MSG\_LIST, 378
  - LBM\_ASYNC\_OP\_TYPE\_RCV\_UMQ\_QUEUE\_MSG\_RETRIEVE, 378
  - lbm\_async\_operation\_cancel, 437
  - LBM\_ASYNC\_OPERATION\_CANCEL\_FLAG\_NONBLOCK, 378
  - lbm\_async\_operation\_function\_cb, 408
  - lbm\_async\_operation\_status, 438
  - LBM\_ASYNC\_OPERATION\_STATUS\_FLAG\_NONBLOCK, 378
  - lbm\_auth\_set\_credentials, 438
  - lbm\_authstorage\_addtpnam, 439
  - lbm\_authstorage\_checkpermission, 439
  - lbm\_authstorage\_close\_storage\_xml, 440
  - lbm\_authstorage\_deltppnam, 440
  - lbm\_authstorage\_load\_roletable, 440
  - lbm\_authstorage\_open\_storage\_xml, 441
  - lbm\_authstorage\_print\_roletable, 441

- lbm\_authstorage\_roletable\_add\_-  
role\_action, 441
- lbm\_authstorage\_unload\_roletable,  
442
- lbm\_authstorage\_user\_add\_role,  
442
- lbm\_authstorage\_user\_del\_role, 442
- lbm\_cancel\_fd, 443
- lbm\_cancel\_fd\_ex, 443
- lbm\_cancel\_timer, 444
- lbm\_cancel\_timer\_ex, 445
- LBM\_CHAIN\_ELEM\_APPHDR,  
378
- LBM\_CHAIN\_ELEM\_-  
CHANNEL\_NUMBER,  
379
- LBM\_CHAIN\_ELEM\_GW\_INFO,  
379
- LBM\_CHAIN\_ELEM\_HF\_SQN,  
379
- LBM\_CHAIN\_ELEM\_-  
PROPERTIES\_LENGTH,  
379
- LBM\_CHAIN\_ELEM\_USER\_-  
DATA, 379
- lbm\_config, 445
- lbm\_config\_xml\_file, 445
- lbm\_config\_xml\_string, 446
- lbm\_context\_attr\_create, 446
- lbm\_context\_attr\_create\_default,  
447
- lbm\_context\_attr\_create\_from\_xml,  
447
- lbm\_context\_attr\_delete, 447
- lbm\_context\_attr\_dump, 448
- lbm\_context\_attr\_dup, 448
- lbm\_context\_attr\_getopt, 448
- lbm\_context\_attr\_option\_size, 449
- lbm\_context\_attr\_set\_from\_xml,  
449
- lbm\_context\_attr\_setopt, 449
- lbm\_context\_attr\_str\_getopt, 450
- lbm\_context\_attr\_str\_setopt, 450
- lbm\_context\_create, 451
- lbm\_context\_delete, 451
- lbm\_context\_delete\_ex, 452
- lbm\_context\_dump, 452
- lbm\_context\_event\_cb\_proc, 408
- lbm\_context\_event\_func\_t, 408
- LBM\_CONTEXT\_EVENT\_-  
UMQ\_INSTANCE\_LIST\_-  
NOTIFICATION, 379
- LBM\_CONTEXT\_EVENT\_-  
UMQ\_REGISTRATION\_-  
COMPLETE\_EX, 379
- LBM\_CONTEXT\_EVENT\_-  
UMQ\_REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_-  
QUORUM, 379
- lbm\_context\_event\_umq\_-  
registration\_complete\_ex\_t,  
408
- LBM\_CONTEXT\_EVENT\_-  
UMQ\_REGISTRATION\_-  
ERROR, 379
- lbm\_context\_event\_umq\_-  
registration\_ex\_t, 408
- LBM\_CONTEXT\_EVENT\_-  
UMQ\_REGISTRATION\_-  
SUCCESS\_EX, 380
- lbm\_context\_from\_rcv, 452
- lbm\_context\_from\_src, 452
- lbm\_context\_from\_wildcard\_rcv,  
453
- lbm\_context\_get\_name, 453
- lbm\_context\_getopt, 453
- lbm\_context\_lbtipc\_unblock, 454
- lbm\_context\_process\_events, 454
- lbm\_context\_process\_lbtipc\_-  
messages, 454
- lbm\_context\_rcv\_immediate\_msgs,  
455
- lbm\_context\_rcv\_immediate\_-  
msgs\_func\_t, 409
- lbm\_context\_rcv\_immediate\_-  
topic\_msgs, 455
- lbm\_context\_reactor\_only\_create,  
456
- lbm\_context\_reset\_im\_rcv\_-  
transport\_stats, 456
- lbm\_context\_reset\_im\_src\_-  
transport\_stats, 457

- lbm\_context\_reset\_rcv\_transport\_-stats, 457
- lbm\_context\_reset\_src\_transport\_-stats, 457
- lbm\_context\_reset\_stats, 457
- lbm\_context\_retrieve\_im\_rcv\_-transport\_stats, 457
- lbm\_context\_retrieve\_im\_src\_-transport\_stats, 458
- lbm\_context\_retrieve\_rcv\_-transport\_stats, 458
- lbm\_context\_retrieve\_src\_-transport\_stats, 459
- lbm\_context\_retrieve\_stats, 459
- lbm\_context\_set\_name, 460
- lbm\_context\_setopt, 460
- lbm\_context\_src\_cb\_proc, 409
- lbm\_context\_src\_event\_func\_t, 409
- lbm\_context\_stats\_t, 409
- lbm\_context\_str\_getopt, 460
- lbm\_context\_str\_setopt, 461
- lbm\_context\_topic\_resolution\_-request, 461
- lbm\_context\_unblock, 462
- lbm\_create\_random\_id, 462
- lbm\_ctx\_umq\_get\_inflight, 462
- lbm\_ctx\_umq\_queue\_topic\_list, 463
- lbm\_daemon\_event\_cb\_proc, 410
- LBM\_DAEMON\_EVENT\_-CONNECT\_ERROR, 380
- LBM\_DAEMON\_EVENT\_-CONNECT\_TIMEOUT, 380
- LBM\_DAEMON\_EVENT\_-CONNECTED, 380
- LBM\_DAEMON\_EVENT\_-DISCONNECTED, 380
- lbm\_debug\_dump, 463
- lbm\_debug\_filename, 463
- lbm\_debug\_mask, 464
- lbm\_deserialize\_response, 464
- LBM\_EDAEMONCONN, 380
- LBM\_EINPROGRESS, 380
- LBM\_EINVAL, 380
- LBM\_EMSG\_SELECTOR, 380
- LBM\_ENO\_QUEUE\_REG, 381
- LBM\_ENO\_STORE\_REG, 381
- LBM\_ENOMEM, 381
- LBM\_EOP, 381
- LBM\_EOPNOTSUPP, 381
- LBM\_EOS, 381
- lbm\_errmsg, 464
- lbm\_errnum, 464
- LBM\_ETIMEDOUT, 381
- LBM\_EUMENOREG, 381
- lbm\_event\_dispatch, 465
- lbm\_event\_dispatch\_unblock, 465
- lbm\_event\_queue\_attr\_create, 465
- lbm\_event\_queue\_attr\_create\_-default, 466
- lbm\_event\_queue\_attr\_create\_-from\_xml, 466
- lbm\_event\_queue\_attr\_delete, 466
- lbm\_event\_queue\_attr\_dump, 467
- lbm\_event\_queue\_attr\_dup, 467
- lbm\_event\_queue\_attr\_getopt, 467
- lbm\_event\_queue\_attr\_option\_size, 468
- lbm\_event\_queue\_attr\_set\_from\_-xml, 468
- lbm\_event\_queue\_attr\_setopt, 468
- lbm\_event\_queue\_attr\_str\_getopt, 469
- lbm\_event\_queue\_attr\_str\_setopt, 469
- LBM\_EVENT\_QUEUE\_BLOCK, 381
- lbm\_event\_queue\_cancel\_cb\_proc, 410
- lbm\_event\_queue\_create, 470
- LBM\_EVENT\_QUEUE\_DELAY\_-WARNING, 381
- lbm\_event\_queue\_delete, 470
- lbm\_event\_queue\_dump, 470
- LBM\_EVENT\_QUEUE\_-ENQUEUE\_NOTIFICATION, 382
- lbm\_event\_queue\_from\_rcv, 471
- lbm\_event\_queue\_from\_src, 471
- lbm\_event\_queue\_from\_wildcard\_-rcv, 471
- lbm\_event\_queue\_getopt, 471

- lbm\_event\_queue\_monitor\_proc, 411
- LBM\_EVENT\_QUEUE\_POLL, 382
- lbm\_event\_queue\_reset\_stats, 472
- lbm\_event\_queue\_retrieve\_stats, 472
- lbm\_event\_queue\_setopt, 472
- lbm\_event\_queue\_shutdown, 473
- lbm\_event\_queue\_size, 473
- LBM\_EVENT\_QUEUE\_SIZE\_-WARNING, 382
- lbm\_event\_queue\_stats\_t, 412
- lbm\_event\_queue\_str\_getopt, 473
- lbm\_event\_queue\_str\_setopt, 474
- LBM\_EWOULDBLOCK, 382
- lbm\_fd\_cb\_proc, 412
- LBM\_FD\_EVENT\_ACCEPT, 382
- LBM\_FD\_EVENT\_ALL, 382
- LBM\_FD\_EVENT\_CLOSE, 382
- LBM\_FD\_EVENT\_CONNECT, 382
- LBM\_FD\_EVENT\_EXCEPT, 382
- LBM\_FD\_EVENT\_READ, 382
- LBM\_FD\_EVENT\_WRITE, 383
- lbm\_flight\_size\_set\_inflight\_cb\_proc, 412
- lbm\_flight\_size\_set\_inflight\_ex\_cb\_proc, 413
- LBM\_FLIGHT\_SIZE\_TYPE\_ULB, 383
- LBM\_FLIGHT\_SIZE\_TYPE\_-UME, 383
- LBM\_FLIGHT\_SIZE\_TYPE\_-UMQ, 383
- lbm\_get\_jms\_msg\_id, 474
- lbm\_hf\_rcv\_create, 474
- lbm\_hf\_rcv\_delete, 475
- lbm\_hf\_rcv\_delete\_ex, 475
- lbm\_hf\_rcv\_from\_rcv, 476
- lbm\_hf\_rcv\_topic\_dump, 476
- lbm\_hf\_src\_create, 476
- lbm\_hf\_src\_send, 477
- lbm\_hf\_src\_send\_ex, 478
- lbm\_hf\_src\_send\_rcv\_reset, 479
- lbm\_hf\_src\_sendv, 480
- lbm\_hf\_src\_sendv\_ex, 480
- lbm\_hfx\_attr\_create, 482
- lbm\_hfx\_attr\_create\_default, 482
- lbm\_hfx\_attr\_create\_from\_xml, 482
- lbm\_hfx\_attr\_delete, 483
- lbm\_hfx\_attr\_dump, 483
- lbm\_hfx\_attr\_dup, 483
- lbm\_hfx\_attr\_getopt, 484
- lbm\_hfx\_attr\_option\_size, 484
- lbm\_hfx\_attr\_set\_from\_xml, 484
- lbm\_hfx\_attr\_setopt, 484
- lbm\_hfx\_attr\_str\_getopt, 485
- lbm\_hfx\_attr\_str\_setopt, 485
- lbm\_hfx\_create, 486
- lbm\_hfx\_delete, 486
- lbm\_hfx\_delete\_ex, 486
- lbm\_hfx\_dump, 487
- lbm\_hfx\_getopt, 487
- lbm\_hfx\_rcv\_create, 488
- lbm\_hfx\_rcv\_delete, 488
- lbm\_hfx\_rcv\_delete\_ex, 488
- lbm\_hfx\_rcv\_topic\_dump, 489
- lbm\_hfx\_setopt, 489
- lbm\_hfx\_str\_getopt, 490
- lbm\_hfx\_str\_setopt, 490
- lbm\_immediate\_msg\_cb\_proc, 413
- lbm\_iovec\_t, 413
- lbm\_ipv4\_address\_mask\_t, 413
- lbm\_is\_ume\_capable, 491
- lbm\_is\_umq\_capable, 491
- lbm\_license\_file, 491
- lbm\_license\_str, 491
- lbm\_license\_ummmn\_valid, 492
- lbm\_license\_vds\_valid, 492
- lbm\_log, 492
- LBM\_LOG\_ALERT, 383
- lbm\_log\_cb\_proc, 414
- LBM\_LOG\_CRIT, 383
- LBM\_LOG\_DEBUG, 383
- LBM\_LOG\_EMERG, 383
- LBM\_LOG\_ERR, 383
- LBM\_LOG\_INFO, 383
- LBM\_LOG\_NOTICE, 384
- LBM\_LOG\_WARNING, 384
- lbm\_logf, 492
- lbm\_mim\_unreclass\_func\_t, 414

- lbm\_mim\_unrecloss\_function\_cb, 414
- LBM\_MSG\_BOS, 384
- lbm\_msg\_channel\_info\_t, 415
- LBM\_MSG\_COMPLETE\_BATCH, 384
- LBM\_MSG\_DATA, 384
- lbm\_msg\_delete, 492
- LBM\_MSG\_END\_BATCH, 384
- LBM\_MSG\_EOS, 384
- lbm\_msg\_extract\_ume\_ack, 493
- LBM\_MSG\_FLAG\_DELIVERY\_-LATENCY, 384
- LBM\_MSG\_FLAG\_END\_BATCH, 384
- LBM\_MSG\_FLAG\_HF\_32, 385
- LBM\_MSG\_FLAG\_HF\_64, 385
- LBM\_MSG\_FLAG\_HF\_-DUPLICATE, 385
- LBM\_MSG\_FLAG\_HF\_-OPTIONAL, 385
- LBM\_MSG\_FLAG\_HF\_PASS\_-THROUGH, 385
- LBM\_MSG\_FLAG\_IMMEDIATE, 385
- LBM\_MSG\_FLAG\_-NUMBERED\_CHANNEL, 385
- LBM\_MSG\_FLAG\_OTR, 385
- LBM\_MSG\_FLAG\_-RETRANSMIT, 385
- LBM\_MSG\_FLAG\_START\_-BATCH, 385
- LBM\_MSG\_FLAG\_TOPICLESS, 386
- LBM\_MSG\_FLAG\_UME\_-RETRANSMIT, 386
- LBM\_MSG\_FLAG\_UME\_SRC\_-REGID, 386
- LBM\_MSG\_FLAG\_UMQ\_-REASSIGNED, 386
- LBM\_MSG\_FLAG\_UMQ\_-RESUBMITTED, 386
- LBM\_MSG\_FLUSH, 386
- lbm\_msg\_fragment\_info\_t, 415
- lbm\_msg\_gateway\_info\_t, 415
- LBM\_MSG\_HF\_RESET, 386
- LBM\_MSG\_IOV\_GATHER, 386
- lbm\_msg\_is\_fragment, 493
- LBM\_MSG\_NO\_SOURCE\_-NOTIFICATION, 387
- lbm\_msg\_properties\_clear, 493
- lbm\_msg\_properties\_create, 494
- lbm\_msg\_properties\_delete, 494
- lbm\_msg\_properties\_get, 495
- lbm\_msg\_properties\_iter\_create, 495
- lbm\_msg\_properties\_iter\_delete, 496
- lbm\_msg\_properties\_iter\_first, 496
- lbm\_msg\_properties\_iter\_next, 497
- LBM\_MSG\_PROPERTIES\_-MAX\_NAMELEN, 387
- lbm\_msg\_properties\_set, 497
- LBM\_MSG\_PROPERTY\_-BOOLEAN, 387
- LBM\_MSG\_PROPERTY\_BYTE, 387
- LBM\_MSG\_PROPERTY\_-DOUBLE, 387
- LBM\_MSG\_PROPERTY\_FLOAT, 387
- LBM\_MSG\_PROPERTY\_INT, 387
- LBM\_MSG\_PROPERTY\_LONG, 387
- LBM\_MSG\_PROPERTY\_NONE, 387
- LBM\_MSG\_PROPERTY\_SHORT, 388
- LBM\_MSG\_PROPERTY\_STRING, 388
- LBM\_MSG\_REQUEST, 388
- LBM\_MSG\_RESPONSE, 388
- lbm\_msg\_retain, 498
- lbm\_msg\_retrieve\_fragment\_info, 498
- lbm\_msg\_retrieve\_gateway\_info, 498
- lbm\_msg\_retrieve\_msgid, 498
- lbm\_msg\_retrieve\_umq\_index, 499
- LBM\_MSG\_START\_BATCH, 388

- lbm\_msg\_ume\_can\_send\_explicit\_ -  
ack, [499](#)
- LBM\_MSG\_UME\_-  
DEREGISTRATION\_-  
COMPLETE\_EX, [388](#)
- lbm\_msg\_ume\_deregistration\_ex\_t,  
[415](#)
- LBM\_MSG\_UME\_-  
DEREGISTRATION\_-  
SUCCESS\_EX, [388](#)
- LBM\_MSG\_UME\_-  
DEREGISTRATION\_-  
SUCCESS\_EX\_FLAG\_RPP,  
[388](#)
- LBM\_MSG\_UME\_-  
REGISTRATION\_CHANGE,  
[388](#)
- LBM\_MSG\_UME\_-  
REGISTRATION\_-  
COMPLETE\_EX, [389](#)
- LBM\_MSG\_UME\_-  
REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_-  
QUORUM, [389](#)
- LBM\_MSG\_UME\_-  
REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_-  
RXREQMAX, [389](#)
- LBM\_MSG\_UME\_-  
REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_-  
SRC\_SID, [389](#)
- lbm\_msg\_ume\_registration\_-  
complete\_ex\_t, [415](#)
- LBM\_MSG\_UME\_-  
REGISTRATION\_ERROR,  
[389](#)
- lbm\_msg\_ume\_registration\_ex\_t,  
[416](#)
- LBM\_MSG\_UME\_-  
REGISTRATION\_SUCCESS,  
[389](#)
- LBM\_MSG\_UME\_-  
REGISTRATION\_-  
SUCCESS\_EX, [389](#)
- LBM\_MSG\_UME\_-  
REGISTRATION\_-  
SUCCESS\_EX\_FLAG\_-  
NOCACHE, [389](#)
- LBM\_MSG\_UME\_-  
REGISTRATION\_-  
SUCCESS\_EX\_FLAG\_OLD,  
[390](#)
- LBM\_MSG\_UME\_-  
REGISTRATION\_-  
SUCCESS\_EX\_FLAG\_RPP,  
[390](#)
- LBM\_MSG\_UME\_-  
REGISTRATION\_-  
SUCCESS\_EX\_FLAG\_-  
SRC\_SID, [390](#)
- lbm\_msg\_ume\_registration\_t, [416](#)
- lbm\_msg\_ume\_send\_explicit\_ack,  
[499](#)
- LBM\_MSG\_UMQ\_-  
DEREGISTRATION\_-  
COMPLETE\_EX, [390](#)
- LBM\_MSG\_UMQ\_-  
DEREGISTRATION\_-  
COMPLETE\_EX\_FLAG\_-  
ULB, [390](#)
- lbm\_msg\_umq\_deregistration\_-  
complete\_ex\_t, [416](#)
- LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNED\_EX, [390](#)
- LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNED\_EX\_FLAG\_-  
REQUESTED, [390](#)
- LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNED\_EX\_FLAG\_ULB,  
[390](#)
- lbm\_msg\_umq\_index\_assigned\_-  
ex\_t, [416](#)
- LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNMENT\_-  
ELIGIBILITY\_ERROR,  
[391](#)
- LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNMENT\_-  
ELIGIBILITY\_START\_-  
COMPLETE\_EX, [391](#)

- LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNMENT\_-  
ELIGIBILITY\_START\_-  
COMPLETE\_EX\_FLAG\_-  
ULB, 391
- lbm\_msg\_umq\_index\_assignment\_-  
eligibility\_start\_complete\_ex\_t,  
416
- LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNMENT\_-  
ELIGIBILITY\_STOP\_-  
COMPLETE\_EX, 391
- LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNMENT\_-  
ELIGIBILITY\_STOP\_-  
COMPLETE\_EX\_FLAG\_-  
ULB, 391
- lbm\_msg\_umq\_index\_assignment\_-  
eligibility\_stop\_complete\_ex\_t,  
416
- LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNMENT\_ERROR,  
391
- LBM\_MSG\_UMQ\_INDEX\_-  
RELEASED\_EX, 391
- LBM\_MSG\_UMQ\_INDEX\_-  
RELEASED\_EX\_FLAG\_-  
ULB, 391
- lbm\_msg\_umq\_index\_released\_ex\_-  
t, 416
- lbm\_msg\_umq\_reassign, 500
- LBM\_MSG\_UMQ\_REASSIGN\_-  
FLAG\_DISCARD, 392
- LBM\_MSG\_UMQ\_-  
REGISTRATION\_-  
COMPLETE\_EX, 392
- LBM\_MSG\_UMQ\_-  
REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_-  
QUORUM, 392
- LBM\_MSG\_UMQ\_-  
REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_-  
ULB, 392
- lbm\_msg\_umq\_registration\_-  
complete\_ex\_t, 417
- LBM\_MSG\_UMQ\_-  
REGISTRATION\_ERROR,  
392
- LBM\_MSG\_-  
UNRECOVERABLE\_LOSS,  
392
- LBM\_MSG\_-  
UNRECOVERABLE\_LOSS\_-  
BURST, 392
- lbm\_multicast\_immediate\_message,  
500
- lbm\_multicast\_immediate\_request,  
500
- lbm\_queue\_immediate\_message,  
501
- LBM\_RCV\_BLOCK, 392
- LBM\_RCV\_BLOCK\_TEMP, 392
- lbm\_rcv\_cb\_proc, 417
- lbm\_rcv\_create, 502
- lbm\_rcv\_delete, 502
- lbm\_rcv\_delete\_ex, 503
- lbm\_rcv\_from\_hf\_rcv, 503
- lbm\_rcv\_from\_hfx\_rcv, 504
- lbm\_rcv\_getopt, 504
- lbm\_rcv\_msg\_source\_clientd, 504
- LBM\_RCV\_NONBLOCK, 393
- lbm\_rcv\_reset\_all\_transport\_stats,  
504
- lbm\_rcv\_reset\_transport\_stats, 505
- lbm\_rcv\_retrieve\_all\_transport\_-  
stats, 505
- lbm\_rcv\_retrieve\_transport\_stats,  
505
- lbm\_rcv\_setopt, 506
- lbm\_rcv\_src\_notification\_create\_-  
function\_cb, 417
- lbm\_rcv\_src\_notification\_delete\_-  
function\_cb, 418
- lbm\_rcv\_src\_notification\_func\_t,  
418
- lbm\_rcv\_str\_getopt, 506
- lbm\_rcv\_str\_setopt, 507
- lbm\_rcv\_subscribe\_channel, 507
- lbm\_rcv\_topic\_attr\_create, 508
- lbm\_rcv\_topic\_attr\_create\_default,  
508

- lbm\_rcv\_topic\_attr\_create\_from\_xml, 508
- lbm\_rcv\_topic\_attr\_delete, 509
- lbm\_rcv\_topic\_attr\_dump, 509
- lbm\_rcv\_topic\_attr\_dup, 509
- lbm\_rcv\_topic\_attr\_getopt, 510
- lbm\_rcv\_topic\_attr\_option\_size, 510
- lbm\_rcv\_topic\_attr\_set\_from\_xml, 510
- lbm\_rcv\_topic\_attr\_setopt, 511
- lbm\_rcv\_topic\_attr\_str\_getopt, 511
- lbm\_rcv\_topic\_attr\_str\_setopt, 512
- lbm\_rcv\_topic\_dump, 512
- lbm\_rcv\_topic\_lookup, 512
- LBM\_RCV\_TOPIC\_STATS\_FLAG\_SRC\_VALID, 393
- lbm\_rcv\_topic\_stats\_t, 418
- lbm\_rcv\_transport\_stats\_daemon\_t, 418
- lbm\_rcv\_transport\_stats\_t, 419
- lbm\_rcv\_ume\_deregister, 513
- lbm\_rcv\_umq\_deregister, 513
- lbm\_rcv\_umq\_index\_release, 513
- lbm\_rcv\_umq\_index\_reserve, 514
- lbm\_rcv\_umq\_index\_start\_assignment, 514
- lbm\_rcv\_umq\_index\_stop\_assignment, 514
- lbm\_rcv\_umq\_queue\_msg\_list, 515
- lbm\_rcv\_umq\_queue\_msg\_retrieve, 515
- lbm\_rcv\_unsubscribe\_channel, 516
- lbm\_rcv\_unsubscribe\_channel\_ex, 517
- lbm\_register\_fd, 517
- lbm\_request\_cb\_proc, 419
- lbm\_request\_delete, 518
- lbm\_request\_delete\_ex, 518
- lbm\_response\_delete, 518
- lbm\_schedule\_timer, 519
- lbm\_schedule\_timer\_recurring, 520
- lbm\_send\_request, 520
- lbm\_send\_request\_ex, 521
- lbm\_send\_response, 522
- lbm\_serialize\_response, 522
- lbm\_serialized\_response\_delete, 523
- lbm\_set\_lbtrm\_loss\_rate, 523
- lbm\_set\_lbtrm\_src\_loss\_rate, 523
- lbm\_set\_lbtru\_loss\_rate, 523
- lbm\_set\_lbtru\_src\_loss\_rate, 524
- lbm\_set\_uim\_loss\_rate, 524
- lbm\_set\_umm\_info, 524
- LBM\_SRC\_BLOCK, 393
- LBM\_SRC\_BLOCK\_TEMP, 393
- lbm\_src\_buff\_acquire, 524
- lbm\_src\_buffs\_cancel, 525
- lbm\_src\_buffs\_complete, 525
- lbm\_src\_buffs\_complete\_and\_acquire, 526
- lbm\_src\_cb\_proc, 419
- lbm\_src\_channel\_create, 526
- lbm\_src\_channel\_delete, 526
- lbm\_src\_cost\_function\_cb, 421
- LBM\_SRC\_COST\_FUNCTION\_REJECT, 393
- lbm\_src\_create, 527
- lbm\_src\_delete, 527
- lbm\_src\_delete\_ex, 528
- LBM\_SRC\_EVENT\_CONNECT, 393
- LBM\_SRC\_EVENT\_DAEMON\_CONFIRM, 393
- LBM\_SRC\_EVENT\_DISCONNECT, 394
- LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION, 394
- LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION\_STATE\_OVER, 394
- LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION\_STATE\_UNDER, 394
- lbm\_src\_event\_flight\_size\_notification\_t, 421
- LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION\_TYPE\_ULB, 394
- LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION\_TYPE\_UME, 394
- LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_NOTIFICATION\_

- TYPE\_UMQ, 394
- LBM\_SRC\_EVENT\_-  
SEQUENCE\_NUMBER\_-  
INFO, 395
- lbm\_src\_event\_sequence\_number\_-  
info\_t, 421
- lbm\_src\_event\_ume\_ack\_ex\_info\_t,  
422
- lbm\_src\_event\_ume\_ack\_info\_t,  
422
- LBM\_SRC\_EVENT\_-  
UME\_DELIVERY\_-  
CONFIRMATION, 395
- LBM\_SRC\_EVENT\_-  
UME\_DELIVERY\_-  
CONFIRMATION\_EX, 395
- LBM\_SRC\_EVENT\_-  
UME\_DELIVERY\_-  
CONFIRMATION\_EX\_-  
FLAG\_EXACK, 395
- LBM\_SRC\_EVENT\_-  
UME\_DELIVERY\_-  
CONFIRMATION\_EX\_-  
FLAG\_OOD, 395
- LBM\_SRC\_EVENT\_-  
UME\_DELIVERY\_-  
CONFIRMATION\_EX\_-  
FLAG\_UNIQUEACKS, 395
- LBM\_SRC\_EVENT\_-  
UME\_DELIVERY\_-  
CONFIRMATION\_EX\_-  
FLAG\_UREGID, 395
- LBM\_SRC\_EVENT\_-  
UME\_DELIVERY\_-  
CONFIRMATION\_EX\_-  
FLAG\_WHOLE\_MESSAGE\_-  
CONFIRMED, 396
- LBM\_SRC\_EVENT\_UME\_-  
DEREGISTRATION\_-  
COMPLETE\_EX, 396
- lbm\_src\_event\_ume\_-  
deregistration\_ex\_t, 422
- LBM\_SRC\_EVENT\_UME\_-  
DEREGISTRATION\_-  
SUCCESS\_EX, 396
- LBM\_SRC\_EVENT\_UME\_-  
MESSAGE\_NOT\_STABLE,  
396
- LBM\_SRC\_EVENT\_UME\_-  
MESSAGE\_NOT\_STABLE\_-  
FLAG\_LOSS, 396
- LBM\_SRC\_EVENT\_UME\_-  
MESSAGE\_NOT\_STABLE\_-  
FLAG\_STORE, 396
- LBM\_SRC\_EVENT\_UME\_-  
MESSAGE\_NOT\_STABLE\_-  
FLAG\_TIMEOUT, 396
- LBM\_SRC\_EVENT\_UME\_-  
MESSAGE\_RECLAIMED,  
397
- LBM\_SRC\_EVENT\_UME\_-  
MESSAGE\_RECLAIMED\_-  
EX, 397
- LBM\_SRC\_EVENT\_UME\_-  
MESSAGE\_RECLAIMED\_-  
EX\_FLAG\_FORCED, 397
- LBM\_SRC\_EVENT\_UME\_-  
MESSAGE\_STABLE, 397
- LBM\_SRC\_EVENT\_UME\_-  
MESSAGE\_STABLE\_EX,  
397
- LBM\_SRC\_EVENT\_UME\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_INTERGROUP\_-  
STABLE, 397
- LBM\_SRC\_EVENT\_UME\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_INTRAGROUP\_-  
STABLE, 397
- LBM\_SRC\_EVENT\_UME\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_STABLE, 397
- LBM\_SRC\_EVENT\_UME\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_STORE, 398
- LBM\_SRC\_EVENT\_UME\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_USER, 398
- LBM\_SRC\_EVENT\_UME\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_WHOLE\_MESSAGE\_-

- STABLE, 398
- LBM\_SRC\_EVENT\_UME\_-  
REGISTRATION\_-  
COMPLETE\_EX, 398
- LBM\_SRC\_EVENT\_UME\_-  
REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_-  
QUORUM, 398
- lbm\_src\_event\_ume\_registration\_-  
complete\_ex\_t, 422
- LBM\_SRC\_EVENT\_UME\_-  
REGISTRATION\_ERROR,  
398
- lbm\_src\_event\_ume\_registration\_-  
ex\_t, 422
- LBM\_SRC\_EVENT\_UME\_-  
REGISTRATION\_SUCCESS,  
398
- LBM\_SRC\_EVENT\_UME\_-  
REGISTRATION\_-  
SUCCESS\_EX, 398
- LBM\_SRC\_EVENT\_UME\_-  
REGISTRATION\_-  
SUCCESS\_EX\_FLAG\_-  
NOACKS, 399
- LBM\_SRC\_EVENT\_UME\_-  
REGISTRATION\_-  
SUCCESS\_EX\_FLAG\_OLD,  
399
- LBM\_SRC\_EVENT\_UME\_-  
REGISTRATION\_-  
SUCCESS\_EX\_FLAG\_RPP,  
399
- lbm\_src\_event\_ume\_registration\_t,  
422
- LBM\_SRC\_EVENT\_UME\_-  
STORE\_UNRESPONSIVE,  
399
- LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_ID\_INFO, 399
- lbm\_src\_event\_umq\_message\_id\_-  
info\_t, 423
- LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_STABLE\_EX,  
399
- LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_INTERGROUP\_-  
STABLE, 399
- LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_INTRAGROUP\_-  
STABLE, 400
- LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_STABLE, 400
- LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_USER, 400
- LBM\_SRC\_EVENT\_UMQ\_-  
REGISTRATION\_-  
COMPLETE\_EX, 400
- LBM\_SRC\_EVENT\_UMQ\_-  
REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_-  
QUORUM, 400
- lbm\_src\_event\_umq\_registration\_-  
complete\_ex\_t, 423
- LBM\_SRC\_EVENT\_UMQ\_-  
REGISTRATION\_ERROR,  
400
- lbm\_src\_event\_umq\_stability\_ack\_-  
info\_ex\_t, 423
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_ASSIGNED\_EX,  
400
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_COMPLETE\_EX,  
400
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_CONSUMED\_-  
EX, 401
- lbm\_src\_event\_umq\_ulb\_message\_-  
info\_ex\_t, 423
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_REASSIGNED\_-  
EX, 401
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_REASSIGNED\_-  
EX\_FLAG\_EXPLICIT, 401

- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_TIMEOUT\_EX,  
401
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_TIMEOUT\_EX\_-  
FLAG\_DISCARD, 401
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_TIMEOUT\_EX\_-  
FLAG\_EXPLICIT, 401
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_TIMEOUT\_EX\_-  
FLAG\_MAX\_REASSIGNS,  
401
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_TIMEOUT\_EX\_-  
FLAG\_TOTAL\_LIFETIME\_-  
EXPIRED, 402
- LBM\_SRC\_EVENT\_UMQ\_-  
ULB\_RECEIVER\_-  
DEREGISTRATION\_EX,  
402
- lbm\_src\_event\_umq\_ulb\_receiver\_-  
info\_ex\_t, 423
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
RECEIVER\_READY\_EX,  
402
- LBM\_SRC\_EVENT\_UMQ\_-  
ULB\_RECEIVER\_-  
REGISTRATION\_EX, 402
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
RECEIVER\_TIMEOUT\_EX,  
402
- LBM\_SRC\_EVENT\_WAKEUP,  
402
- LBM\_SRC\_EVENT\_WAKEUP\_-  
FLAG\_MIM, 402
- LBM\_SRC\_EVENT\_WAKEUP\_-  
FLAG\_NORMAL, 403
- LBM\_SRC\_EVENT\_WAKEUP\_-  
FLAG\_REQUEST, 403
- LBM\_SRC\_EVENT\_WAKEUP\_-  
FLAG\_RESPONSE, 403
- LBM\_SRC\_EVENT\_WAKEUP\_-  
FLAG\_UIM, 403
- lbm\_src\_event\_wakeup\_t, 423
- lbm\_src\_flush, 528
- lbm\_src\_get\_inflight, 528
- lbm\_src\_get\_inflight\_ex, 529
- lbm\_src\_getopt, 529
- LBM\_SRC\_NONBLOCK, 403
- lbm\_src\_notify\_func\_t, 423
- lbm\_src\_notify\_function\_cb, 424
- lbm\_src\_reset\_transport\_stats, 530
- lbm\_src\_retrieve\_transport\_stats,  
530
- lbm\_src\_send, 530
- lbm\_src\_send\_ex, 531
- LBM\_SRC\_SEND\_EX\_FLAG\_-  
APPHDR\_CHAIN, 403
- LBM\_SRC\_SEND\_EX\_FLAG\_-  
CHANNEL, 403
- LBM\_SRC\_SEND\_EX\_FLAG\_-  
HF\_32, 403
- LBM\_SRC\_SEND\_EX\_FLAG\_-  
HF\_64, 403
- LBM\_SRC\_SEND\_EX\_FLAG\_-  
HF\_OPTIONAL, 404
- LBM\_SRC\_SEND\_EX\_FLAG\_-  
PROPERTIES, 404
- LBM\_SRC\_SEND\_EX\_FLAG\_-  
SEQUENCE\_NUMBER\_-  
INFO, 404
- LBM\_SRC\_SEND\_EX\_FLAG\_-  
SEQUENCE\_NUMBER\_-  
INFO\_FRAGONLY, 404
- LBM\_SRC\_SEND\_EX\_FLAG\_-  
UME\_CLIENTD, 404
- LBM\_SRC\_SEND\_EX\_FLAG\_-  
UMQ\_CLIENTD, 404
- LBM\_SRC\_SEND\_EX\_FLAG\_-  
UMQ\_INDEX, 404
- LBM\_SRC\_SEND\_EX\_FLAG\_-  
UMQ\_MESSAGE\_ID\_INFO,  
404
- LBM\_SRC\_SEND\_EX\_FLAG\_-  
UMQ\_TOTAL\_LIFETIME,  
404
- lbm\_src\_send\_ex\_info\_t, 424
- lbm\_src\_sendv, 532
- lbm\_src\_sendv\_ex, 533
- lbm\_src\_setopt, 534
- lbm\_src\_str\_getopt, 534

- lbm\_src\_str\_setopt, [534](#)
- lbm\_src\_topic\_alloc, [535](#)
- lbm\_src\_topic\_attr\_create, [535](#)
- lbm\_src\_topic\_attr\_create\_default, [536](#)
- lbm\_src\_topic\_attr\_create\_from\_xml, [536](#)
- lbm\_src\_topic\_attr\_delete, [537](#)
- lbm\_src\_topic\_attr\_dump, [537](#)
- lbm\_src\_topic\_attr\_dup, [537](#)
- lbm\_src\_topic\_attr\_getopt, [537](#)
- lbm\_src\_topic\_attr\_option\_size, [538](#)
- lbm\_src\_topic\_attr\_set\_from\_xml, [538](#)
- lbm\_src\_topic\_attr\_setopt, [538](#)
- lbm\_src\_topic\_attr\_str\_getopt, [539](#)
- lbm\_src\_topic\_attr\_str\_setopt, [539](#)
- lbm\_src\_topic\_dump, [540](#)
- lbm\_src\_transport\_stats\_daemon\_t, [424](#)
- lbm\_src\_transport\_stats\_t, [424](#)
- lbm\_src\_ume\_deregister, [540](#)
- lbm\_str\_hash\_func\_ex\_t, [424](#)
- lbm\_str\_hash\_function\_cb, [425](#)
- lbm\_str\_hash\_function\_cb\_ex, [425](#)
- lbm\_timer\_cb\_proc, [425](#)
- lbm\_timeval\_t, [426](#)
- lbm\_topic\_from\_src, [540](#)
- LBM\_TOPIC\_RES\_REQUEST\_-ADVERTISEMENT, [405](#)
- LBM\_TOPIC\_RES\_-REQUEST\_CONTEXT\_-ADVERTISEMENT, [405](#)
- LBM\_TOPIC\_RES\_REQUEST\_-CONTEXT\_QUERY, [405](#)
- LBM\_TOPIC\_RES\_REQUEST\_-GW\_REMOTE\_INTEREST, [405](#)
- LBM\_TOPIC\_RES\_REQUEST\_-QUERY, [405](#)
- LBM\_TOPIC\_RES\_REQUEST\_-RESERVED1, [405](#)
- LBM\_TOPIC\_RES\_REQUEST\_-WILDCARD\_QUERY, [405](#)
- lbm\_transport\_source\_format, [540](#)
- lbm\_transport\_source\_info\_t, [426](#)
- lbm\_transport\_source\_parse, [541](#)
- LBM\_TRANSPORT\_STAT\_-DAEMON, [405](#)
- LBM\_TRANSPORT\_STAT\_-LBTIPC, [406](#)
- LBM\_TRANSPORT\_STAT\_-LBTRDMA, [406](#)
- LBM\_TRANSPORT\_STAT\_-LBTRM, [406](#)
- LBM\_TRANSPORT\_STAT\_-LBTRU, [406](#)
- LBM\_TRANSPORT\_STAT\_-LBTSMX, [406](#)
- LBM\_TRANSPORT\_STAT\_TCP, [406](#)
- LBM\_TRANSPORT\_TYPE\_-LBTIPC, [406](#)
- LBM\_TRANSPORT\_TYPE\_-LBTRDMA, [406](#)
- LBM\_TRANSPORT\_TYPE\_-LBTRM, [407](#)
- LBM\_TRANSPORT\_TYPE\_-LBTRU, [407](#)
- LBM\_TRANSPORT\_TYPE\_-LBTSMX, [407](#)
- LBM\_TRANSPORT\_TYPE\_TCP, [407](#)
- lbm\_ucast\_resolver\_entry\_t, [426](#)
- lbm\_ume\_ack\_delete, [541](#)
- lbm\_ume\_ack\_send\_explicit\_ack, [541](#)
- lbm\_ume\_ctx\_rcv\_ctx\_-notification\_create\_function\_cb, [427](#)
- lbm\_ume\_ctx\_rcv\_ctx\_-notification\_delete\_function\_cb, [427](#)
- lbm\_ume\_ctx\_rcv\_ctx\_-notification\_func\_t, [428](#)
- lbm\_ume\_rcv\_recovery\_info\_ex\_-func\_info\_t, [428](#)
- lbm\_ume\_rcv\_recovery\_info\_ex\_-func\_t, [428](#)
- lbm\_ume\_rcv\_recovery\_info\_ex\_-function\_cb, [428](#)

- lbm\_ume\_rcv\_regid\_ex\_func\_info\_t, 429
- lbm\_ume\_rcv\_regid\_ex\_func\_t, 429
- lbm\_ume\_rcv\_regid\_ex\_function\_cb, 429
- lbm\_ume\_rcv\_regid\_func\_t, 429
- lbm\_ume\_rcv\_regid\_function\_cb, 429
- lbm\_ume\_src\_force\_reclaim\_func\_t, 430
- lbm\_ume\_src\_force\_reclaim\_function\_cb, 430
- lbm\_ume\_src\_msg\_stable, 542
- lbm\_ume\_store\_entry\_t, 431
- lbm\_ume\_store\_group\_entry\_t, 431
- lbm\_ume\_store\_name\_entry\_t, 431
- LBM\_UMM\_INFO\_FLAGS\_USE\_SSL, 407
- lbm\_umq\_ctx\_msg\_stable, 542
- LBM\_UMQ\_INDEX\_FLAG\_NUMERIC, 407
- lbm\_umq\_index\_info\_t, 431
- lbm\_umq\_msg\_selector\_create, 542
- lbm\_umq\_msg\_selector\_delete, 543
- lbm\_umq\_msg\_total\_lifetime\_info\_t, 431
- lbm\_umq\_msgid\_t, 431
- lbm\_umq\_queue\_entry\_t, 431
- lbm\_umq\_regid\_t, 431
- lbm\_umq\_ulb\_application\_set\_attr\_t, 432
- lbm\_umq\_ulb\_receiver\_type\_attr\_t, 432
- lbm\_umq\_ulb\_receiver\_type\_entry\_t, 432
- lbm\_unicast\_immediate\_message, 543
- lbm\_unicast\_immediate\_request, 543
- lbm\_version, 544
- lbm\_wildcard\_rcv\_attr\_create, 544
- lbm\_wildcard\_rcv\_attr\_create\_default, 545
- lbm\_wildcard\_rcv\_attr\_create\_from\_xml, 545
- lbm\_wildcard\_rcv\_attr\_delete, 546
- lbm\_wildcard\_rcv\_attr\_dump, 546
- lbm\_wildcard\_rcv\_attr\_dup, 546
- lbm\_wildcard\_rcv\_attr\_getopt, 547
- lbm\_wildcard\_rcv\_attr\_option\_size, 547
- lbm\_wildcard\_rcv\_attr\_set\_from\_xml, 547
- lbm\_wildcard\_rcv\_attr\_setopt, 548
- lbm\_wildcard\_rcv\_attr\_str\_getopt, 548
- lbm\_wildcard\_rcv\_attr\_str\_setopt, 549
- lbm\_wildcard\_rcv\_compare\_func\_t, 432
- lbm\_wildcard\_rcv\_compare\_function\_cb, 432
- lbm\_wildcard\_rcv\_create, 549
- lbm\_wildcard\_rcv\_create\_func\_t, 433
- lbm\_wildcard\_rcv\_create\_function\_cb, 433
- lbm\_wildcard\_rcv\_delete, 550
- lbm\_wildcard\_rcv\_delete\_ex, 550
- lbm\_wildcard\_rcv\_delete\_func\_t, 433
- lbm\_wildcard\_rcv\_delete\_function\_cb, 433
- lbm\_wildcard\_rcv\_dump, 551
- lbm\_wildcard\_rcv\_getopt, 551
- LBM\_WILDCARD\_RCV\_PATTERN\_TYPE\_APP\_CB, 407
- LBM\_WILDCARD\_RCV\_PATTERN\_TYPE\_PCRE, 407
- LBM\_WILDCARD\_RCV\_PATTERN\_TYPE\_REGEX, 407
- lbm\_wildcard\_rcv\_setopt, 551
- lbm\_wildcard\_rcv\_stats\_t, 434
- lbm\_wildcard\_rcv\_str\_getopt, 552
- lbm\_wildcard\_rcv\_str\_setopt, 552
- lbm\_wildcard\_rcv\_subscribe\_channel, 553
- lbm\_wildcard\_rcv\_umq\_deregister, 553

- lbm\_wildcard\_rcv\_umq\_index\_-  
release, [553](#)
- lbm\_wildcard\_rcv\_umq\_index\_-  
start\_assignment, [554](#)
- lbm\_wildcard\_rcv\_umq\_index\_-  
stop\_assignment, [554](#)
- lbm\_wildcard\_rcv\_unsubscribe\_-  
channel, [554](#)
- lbm\_wildcard\_rcv\_unsubscribe\_-  
channel\_ex, [555](#)
- lbm\_win32\_static\_thread\_attach,  
[555](#)
- lbm\_win32\_static\_thread\_detach,  
[555](#)
- lbm\_wrcv\_ume\_deregister, [556](#)  
ume\_liveness\_receiving\_context\_t,  
[434](#)
- lbm\_apphdr\_chain\_append\_elem  
lbm.h, [434](#)
- lbm\_apphdr\_chain\_create  
lbm.h, [435](#)
- lbm\_apphdr\_chain\_delete  
lbm.h, [435](#)
- lbm\_apphdr\_chain\_elem\_t\_stct, [113](#)  
data, [113](#)  
len, [113](#)  
subtype, [113](#)  
type, [114](#)
- lbm\_apphdr\_chain\_iter\_create  
lbm.h, [435](#)
- lbm\_apphdr\_chain\_iter\_create\_from\_-  
msg  
lbm.h, [435](#)
- lbm\_apphdr\_chain\_iter\_current  
lbm.h, [436](#)
- lbm\_apphdr\_chain\_iter\_delete  
lbm.h, [436](#)
- lbm\_apphdr\_chain\_iter\_done  
lbm.h, [436](#)
- lbm\_apphdr\_chain\_iter\_first  
lbm.h, [437](#)
- lbm\_apphdr\_chain\_iter\_next  
lbm.h, [437](#)
- LBM\_ASYNC\_OP\_INFO\_FLAG\_-  
FIRST  
lbm.h, [377](#)
- LBM\_ASYNC\_OP\_INFO\_FLAG\_-  
INLINE  
lbm.h, [377](#)
- LBM\_ASYNC\_OP\_INFO\_FLAG\_LAST  
lbm.h, [377](#)
- LBM\_ASYNC\_OP\_INFO\_FLAG\_-  
ONLY  
lbm.h, [377](#)
- LBM\_ASYNC\_OP\_INVALID\_-  
HANDLE  
lbm.h, [377](#)
- LBM\_ASYNC\_OP\_STATUS\_-  
CANCELED  
lbm.h, [377](#)
- LBM\_ASYNC\_OP\_STATUS\_-  
COMPLETE  
lbm.h, [377](#)
- LBM\_ASYNC\_OP\_STATUS\_ERROR  
lbm.h, [378](#)
- LBM\_ASYNC\_OP\_STATUS\_IN\_-  
PROGRESS  
lbm.h, [378](#)
- LBM\_ASYNC\_OP\_TYPE\_CTX\_-  
UMQ\_QUEUE\_TOPIC\_LIST  
lbm.h, [378](#)
- LBM\_ASYNC\_OP\_TYPE\_RCV\_-  
UMQ\_QUEUE\_MSG\_LIST  
lbm.h, [378](#)
- LBM\_ASYNC\_OP\_TYPE\_RCV\_-  
UMQ\_QUEUE\_MSG\_-  
RETRIEVE  
lbm.h, [378](#)
- lbm\_async\_operation\_cancel  
lbm.h, [437](#)
- LBM\_ASYNC\_OPERATION\_-  
CANCEL\_FLAG\_-  
NONBLOCK  
lbm.h, [378](#)
- lbm\_async\_operation\_func\_t, [115](#)  
clientd, [115](#)  
evq, [115](#)  
flags, [115](#)  
func, [115](#)
- lbm\_async\_operation\_function\_cb  
lbm.h, [408](#)
- lbm\_async\_operation\_info\_t, [116](#)

- flags, [117](#)
- handle, [117](#)
- info, [117](#)
- status, [117](#)
- type, [117](#)
- lbm\_async\_operation\_status
  - lbm.h, [438](#)
- LBM\_ASYNC\_OPERATION\_-  
STATUS\_FLAG\_NONBLOCK
  - lbm.h, [378](#)
- lbm\_auth\_set\_credentials
  - lbm.h, [438](#)
- lbm\_authstorage\_addtpnam
  - lbm.h, [439](#)
- lbm\_authstorage\_checkpermission
  - lbm.h, [439](#)
- lbm\_authstorage\_close\_storage\_xml
  - lbm.h, [440](#)
- lbm\_authstorage\_deltptnam
  - lbm.h, [440](#)
- lbm\_authstorage\_load\_roletable
  - lbm.h, [440](#)
- lbm\_authstorage\_open\_storage\_xml
  - lbm.h, [441](#)
- lbm\_authstorage\_print\_roletable
  - lbm.h, [441](#)
- lbm\_authstorage\_roletable\_add\_role\_  
action
  - lbm.h, [441](#)
- lbm\_authstorage\_unload\_roletable
  - lbm.h, [442](#)
- lbm\_authstorage\_user\_add\_role
  - lbm.h, [442](#)
- lbm\_authstorage\_user\_del\_role
  - lbm.h, [442](#)
- lbm\_cancel\_fd
  - lbm.h, [443](#)
- lbm\_cancel\_fd\_ex
  - lbm.h, [443](#)
- lbm\_cancel\_timer
  - lbm.h, [444](#)
- lbm\_cancel\_timer\_ex
  - lbm.h, [445](#)
- LBM\_CHAIN\_ELEM\_APPHDR
  - lbm.h, [378](#)
- LBM\_CHAIN\_ELEM\_CHANNEL\_  
NUMBER
  - lbm.h, [379](#)
- LBM\_CHAIN\_ELEM\_GW\_INFO
  - lbm.h, [379](#)
- LBM\_CHAIN\_ELEM\_HF\_SQN
  - lbm.h, [379](#)
- LBM\_CHAIN\_ELEM\_PROPERTIES\_  
LENGTH
  - lbm.h, [379](#)
- LBM\_CHAIN\_ELEM\_USER\_DATA
  - lbm.h, [379](#)
- lbm\_config
  - lbm.h, [445](#)
- lbm\_config\_xml\_file
  - lbm.h, [445](#)
- lbm\_config\_xml\_string
  - lbm.h, [446](#)
- lbm\_context\_attr\_create
  - lbm.h, [446](#)
- lbm\_context\_attr\_create\_default
  - lbm.h, [447](#)
- lbm\_context\_attr\_create\_from\_xml
  - lbm.h, [447](#)
- lbm\_context\_attr\_delete
  - lbm.h, [447](#)
- lbm\_context\_attr\_dump
  - lbm.h, [448](#)
- lbm\_context\_attr\_dup
  - lbm.h, [448](#)
- lbm\_context\_attr\_getopt
  - lbm.h, [448](#)
- lbm\_context\_attr\_option\_size
  - lbm.h, [449](#)
- lbm\_context\_attr\_set\_from\_xml
  - lbm.h, [449](#)
- lbm\_context\_attr\_setopt
  - lbm.h, [449](#)
- lbm\_context\_attr\_str\_getopt
  - lbm.h, [450](#)
- lbm\_context\_attr\_str\_setopt
  - lbm.h, [450](#)
- lbm\_context\_create
  - lbm.h, [451](#)
- lbm\_context\_delete
  - lbm.h, [451](#)

- lbm\_context\_delete\_ex
  - lbm.h, 452
- lbm\_context\_dump
  - lbm.h, 452
- lbm\_context\_event\_cb\_proc
  - lbm.h, 408
- lbm\_context\_event\_func\_t
  - lbm.h, 408
- lbm\_context\_event\_func\_t\_stct, 118
- LBM\_CONTEXT\_EVENT\_UMQ\_-
  - INSTANCE\_LIST\_-
  - NOTIFICATION
  - lbm.h, 379
- LBM\_CONTEXT\_EVENT\_UMQ\_-
  - REGISTRATION\_-
  - COMPLETE\_EX
  - lbm.h, 379
- LBM\_CONTEXT\_EVENT\_UMQ\_-
  - REGISTRATION\_-
  - COMPLETE\_EX\_FLAG\_-
  - QUORUM
  - lbm.h, 379
- lbm\_context\_event\_umq\_registration\_-
  - complete\_ex\_t
  - lbm.h, 408
- 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, 379
- lbm\_context\_event\_umq\_registration\_-
  - ex\_t
  - lbm.h, 408
- lbm\_context\_event\_umq\_registration\_-
  - ex\_t\_stct, 121
  - flags, 121
  - queue, 121
  - queue\_id, 121
  - queue\_instance, 121
  - queue\_instance\_index, 121
  - registration\_id, 122
- LBM\_CONTEXT\_EVENT\_UMQ\_-
  - REGISTRATION\_-
  - SUCCESS\_EX
  - lbm.h, 380
- lbm\_context\_from\_rcv
  - lbm.h, 452
- lbm\_context\_from\_src
  - lbm.h, 452
- lbm\_context\_from\_wildcard\_rcv
  - lbm.h, 453
- lbm\_context\_get\_name
  - lbm.h, 453
- lbm\_context\_getopt
  - lbm.h, 453
- lbm\_context\_lbtipc\_unblock
  - lbm.h, 454
- lbm\_context\_process\_events
  - lbm.h, 454
- lbm\_context\_process\_lbtipc\_messages
  - lbm.h, 454
- lbm\_context\_rcv\_immediate\_msgs
  - lbm.h, 455
- lbm\_context\_rcv\_immediate\_msgs\_-
  - func\_t
  - lbm.h, 409
- lbm\_context\_rcv\_immediate\_msgs\_-
  - func\_t\_stct, 123
- lbm\_context\_rcv\_immediate\_topic\_msgs
  - lbm.h, 455
- lbm\_context\_reactor\_only\_create
  - lbm.h, 456
- lbm\_context\_reset\_im\_rcv\_transport\_-
  - stats
  - lbm.h, 456
- lbm\_context\_reset\_im\_src\_transport\_-
  - stats
  - lbm.h, 457
- lbm\_context\_reset\_rcv\_transport\_stats
  - lbm.h, 457
- lbm\_context\_reset\_src\_transport\_stats
  - lbm.h, 457
- lbm\_context\_reset\_stats
  - lbm.h, 457
- lbm\_context\_retrieve\_im\_rcv\_transport\_-
  - stats
  - lbm.h, 457

- lbm\_context\_retrieve\_im\_src\_transport\_ - stats  
lbm.h, 458
- lbm\_context\_retrieve\_rcv\_transport\_stats  
lbm.h, 458
- lbm\_context\_retrieve\_src\_transport\_stats  
lbm.h, 459
- lbm\_context\_retrieve\_stats  
lbm.h, 459
- lbm\_context\_set\_name  
lbm.h, 460
- lbm\_context\_setopt  
lbm.h, 460
- lbm\_context\_src\_cb\_proc  
lbm.h, 409
- lbm\_context\_src\_event\_func\_t  
lbm.h, 409
- lbm\_context\_src\_event\_func\_t\_stct, 124
- lbm\_context\_stats\_t  
lbm.h, 409
- lbm\_context\_stats\_t\_stct, 125
  - fragments\_lost, 126
  - fragments\_unrecoverably\_lost, 126
  - lbrtm\_unknown\_msgs\_rcved, 126
  - lbrtu\_unknown\_msgs\_rcved, 126
  - rcv\_cb\_svc\_time\_max, 126
  - rcv\_cb\_svc\_time\_mean, 127
  - rcv\_cb\_svc\_time\_min, 127
  - resp\_blocked, 127
  - resp\_would\_block, 127
  - send\_blocked, 128
  - send\_would\_block, 128
  - tr\_bytes\_rcved, 128
  - tr\_bytes\_sent, 128
  - tr\_dgrams\_dropped\_malformed, 128
  - tr\_dgrams\_dropped\_type, 128
  - tr\_dgrams\_dropped\_ver, 128
  - tr\_dgrams\_rcved, 129
  - tr\_dgrams\_send\_failed, 129
  - tr\_dgrams\_sent, 129
  - tr\_rcv\_topics, 129
  - tr\_rcv\_unresolved\_topics, 129
  - tr\_src\_topics, 129
  - uim\_dup\_msgs\_rcved, 129
  - uim\_msgs\_no\_stream\_rcved, 130
- lbm\_context\_str\_getopt  
lbm.h, 460
- lbm\_context\_str\_setopt  
lbm.h, 461
- lbm\_context\_topic\_resolution\_request  
lbm.h, 461
- lbm\_context\_unblock  
lbm.h, 462
- lbm\_create\_random\_id  
lbm.h, 462
- lbm\_ctx\_umq\_get\_inflight  
lbm.h, 462
- lbm\_ctx\_umq\_queue\_topic\_list  
lbm.h, 463
- lbm\_ctx\_umq\_queue\_topic\_list\_info\_t,  
131
  - num\_topics, 131
  - topics, 131
- lbm\_daemon\_event\_cb\_proc  
lbm.h, 410
- LBM\_DAEMON\_EVENT\_ - CONNECT\_ERROR  
lbm.h, 380
- LBM\_DAEMON\_EVENT\_ - CONNECT\_TIMEOUT  
lbm.h, 380
- LBM\_DAEMON\_EVENT\_ - CONNECTED  
lbm.h, 380
- LBM\_DAEMON\_EVENT\_ - DISCONNECTED  
lbm.h, 380
- lbm\_debug\_dump  
lbm.h, 463
- lbm\_debug\_filename  
lbm.h, 463
- lbm\_debug\_mask  
lbm.h, 464
- lbm\_delete\_cb\_info\_t\_stct, 132
  - cbproc, 132
  - clientd, 132
- lbm\_delete\_cb\_proc  
lbmht.h, 564
- lbm\_deserialize\_response  
lbm.h, 464
- LBM\_EDAEMONCONN  
lbm.h, 380

- LBM\_EINPROGRESS
  - lbm.h, [380](#)
- LBM\_EINVAL
  - lbm.h, [380](#)
- LBM\_EMMSG\_SELECTOR
  - lbm.h, [380](#)
- LBM\_ENO\_QUEUE\_REG
  - lbm.h, [381](#)
- LBM\_ENO\_STORE\_REG
  - lbm.h, [381](#)
- LBM\_ENOMEM
  - lbm.h, [381](#)
- LBM\_EOP
  - lbm.h, [381](#)
- LBM\_EOPNOTSUPP
  - lbm.h, [381](#)
- LBM\_EOS
  - lbm.h, [381](#)
- lbm\_errmsg
  - lbm.h, [464](#)
- lbm\_errnum
  - lbm.h, [464](#)
- LBM\_ETIMEDOUT
  - lbm.h, [381](#)
- LBM\_EUMENOREG
  - lbm.h, [381](#)
- lbm\_event\_dispatch
  - lbm.h, [465](#)
- lbm\_event\_dispatch\_unblock
  - lbm.h, [465](#)
- lbm\_event\_queue\_attr\_create
  - lbm.h, [465](#)
- lbm\_event\_queue\_attr\_create\_default
  - lbm.h, [466](#)
- lbm\_event\_queue\_attr\_create\_from\_xml
  - lbm.h, [466](#)
- lbm\_event\_queue\_attr\_delete
  - lbm.h, [466](#)
- lbm\_event\_queue\_attr\_dump
  - lbm.h, [467](#)
- lbm\_event\_queue\_attr\_dup
  - lbm.h, [467](#)
- lbm\_event\_queue\_attr\_getopt
  - lbm.h, [467](#)
- lbm\_event\_queue\_attr\_option\_size
  - lbm.h, [468](#)
- lbm\_event\_queue\_attr\_set\_from\_xml
  - lbm.h, [468](#)
- lbm\_event\_queue\_attr\_setopt
  - lbm.h, [468](#)
- lbm\_event\_queue\_attr\_str\_getopt
  - lbm.h, [469](#)
- lbm\_event\_queue\_attr\_str\_setopt
  - lbm.h, [469](#)
- LBM\_EVENT\_QUEUE\_BLOCK
  - lbm.h, [381](#)
- lbm\_event\_queue\_cancel\_cb\_info\_t\_stct,
  - [133](#)
  - cbproc, [133](#)
  - clientd, [133](#)
  - event\_queue, [133](#)
- lbm\_event\_queue\_cancel\_cb\_proc
  - lbm.h, [410](#)
- lbm\_event\_queue\_create
  - lbm.h, [470](#)
- LBM\_EVENT\_QUEUE\_DELAY\_-
  - WARNING
  - lbm.h, [381](#)
- lbm\_event\_queue\_delete
  - lbm.h, [470](#)
- lbm\_event\_queue\_dump
  - lbm.h, [470](#)
- LBM\_EVENT\_QUEUE\_ENQUEUE\_-
  - NOTIFICATION
  - lbm.h, [382](#)
- lbm\_event\_queue\_from\_rcv
  - lbm.h, [471](#)
- lbm\_event\_queue\_from\_src
  - lbm.h, [471](#)
- lbm\_event\_queue\_from\_wildcard\_rcv
  - lbm.h, [471](#)
- lbm\_event\_queue\_getopt
  - lbm.h, [471](#)
- lbm\_event\_queue\_monitor\_proc
  - lbm.h, [411](#)
- LBM\_EVENT\_QUEUE\_POLL
  - lbm.h, [382](#)
- lbm\_event\_queue\_reset\_stats
  - lbm.h, [472](#)
- lbm\_event\_queue\_retrieve\_stats
  - lbm.h, [472](#)
- lbm\_event\_queue\_setopt

- lbm.h, 472
- lbm\_event\_queue\_shutdown
  - lbm.h, 473
- lbm\_event\_queue\_size
  - lbm.h, 473
- LBM\_EVENT\_QUEUE\_SIZE\_ -
  - WARNING
  - lbm.h, 382
- lbm\_event\_queue\_stats\_t
  - lbm.h, 412
- lbm\_event\_queue\_stats\_t\_stct, 134
  - age\_max, 135
  - age\_mean, 135
  - age\_min, 136
  - callback\_events, 136
  - callback\_events\_svc\_max, 136
  - callback\_events\_svc\_mean, 136
  - callback\_events\_svc\_min, 136
  - callback\_events\_tot, 136
  - cancel\_events, 137
  - cancel\_events\_svc\_max, 137
  - cancel\_events\_svc\_mean, 137
  - cancel\_events\_svc\_min, 137
  - cancel\_events\_tot, 137
  - context\_source\_events, 137
  - context\_source\_events\_svc\_max, 137
  - context\_source\_events\_svc\_mean, 138
  - context\_source\_events\_svc\_min, 138
  - context\_source\_events\_tot, 138
  - data\_msgs, 138
  - data\_msgs\_svc\_max, 138
  - data\_msgs\_svc\_mean, 139
  - data\_msgs\_svc\_min, 139
  - data\_msgs\_tot, 139
  - events, 139
  - events\_tot, 139
  - io\_events, 139
  - io\_events\_svc\_max, 139
  - io\_events\_svc\_mean, 140
  - io\_events\_svc\_min, 140
  - io\_events\_tot, 140
  - resp\_msgs, 140
  - resp\_msgs\_svc\_max, 140
  - resp\_msgs\_svc\_mean, 140
  - resp\_msgs\_svc\_min, 141
  - resp\_msgs\_tot, 141
  - source\_events, 141
  - source\_events\_svc\_max, 141
  - source\_events\_svc\_mean, 141
  - source\_events\_svc\_min, 141
  - source\_events\_tot, 142
  - timer\_events, 142
  - timer\_events\_svc\_max, 142
  - timer\_events\_svc\_mean, 142
  - timer\_events\_svc\_min, 142
  - timer\_events\_tot, 142
  - topicless\_im\_msgs, 143
  - topicless\_im\_msgs\_svc\_max, 143
  - topicless\_im\_msgs\_svc\_mean, 143
  - topicless\_im\_msgs\_svc\_min, 143
  - topicless\_im\_msgs\_tot, 143
  - unblock\_events, 144
  - unblock\_events\_tot, 144
  - wrcv\_msgs, 144
  - wrcv\_msgs\_svc\_max, 144
  - wrcv\_msgs\_svc\_mean, 144
  - wrcv\_msgs\_svc\_min, 144
  - wrcv\_msgs\_tot, 145
- lbm\_event\_queue\_str\_getopt
  - lbm.h, 473
- lbm\_event\_queue\_str\_setopt
  - lbm.h, 474
- LBM\_EWOULDBLOCK
  - lbm.h, 382
- lbm\_fd\_cb\_proc
  - lbm.h, 412
- LBM\_FD\_EVENT\_ACCEPT
  - lbm.h, 382
- LBM\_FD\_EVENT\_ALL
  - lbm.h, 382
- LBM\_FD\_EVENT\_CLOSE
  - lbm.h, 382
- LBM\_FD\_EVENT\_CONNECT
  - lbm.h, 382
- LBM\_FD\_EVENT\_EXCEPT
  - lbm.h, 382
- LBM\_FD\_EVENT\_READ
  - lbm.h, 382
- LBM\_FD\_EVENT\_WRITE

- lbm.h, 383
- lbm\_flight\_size\_inflight\_t\_stct, 146
  - bytes, 146
  - messages, 146
- lbm\_flight\_size\_set\_inflight\_cb\_proc
  - lbm.h, 412
- lbm\_flight\_size\_set\_inflight\_ex\_cb\_proc
  - lbm.h, 413
- LBM\_FLIGHT\_SIZE\_TYPE\_ULB
  - lbm.h, 383
- LBM\_FLIGHT\_SIZE\_TYPE\_UME
  - lbm.h, 383
- LBM\_FLIGHT\_SIZE\_TYPE\_UMQ
  - lbm.h, 383
- lbm\_get\_jms\_msg\_id
  - lbm.h, 474
- lbm\_hf\_rcv\_create
  - lbm.h, 474
- lbm\_hf\_rcv\_delete
  - lbm.h, 475
- lbm\_hf\_rcv\_delete\_ex
  - lbm.h, 475
- lbm\_hf\_rcv\_from\_rcv
  - lbm.h, 476
- lbm\_hf\_rcv\_topic\_dump
  - lbm.h, 476
- lbm\_hf\_sequence\_number\_t\_stct, 147
  - u32, 147
  - u64, 147
- lbm\_hf\_src\_create
  - lbm.h, 476
- lbm\_hf\_src\_send
  - lbm.h, 477
- lbm\_hf\_src\_send\_ex
  - lbm.h, 478
- lbm\_hf\_src\_send\_rcv\_reset
  - lbm.h, 479
- lbm\_hf\_src\_sendv
  - lbm.h, 480
- lbm\_hf\_src\_sendv\_ex
  - lbm.h, 480
- lbm\_hfx\_attr\_create
  - lbm.h, 482
- lbm\_hfx\_attr\_create\_default
  - lbm.h, 482
- lbm\_hfx\_attr\_create\_from\_xml
  - lbm.h, 482
- lbm\_hfx\_attr\_delete
  - lbm.h, 483
- lbm\_hfx\_attr\_dump
  - lbm.h, 483
- lbm\_hfx\_attr\_dup
  - lbm.h, 483
- lbm\_hfx\_attr\_getopt
  - lbm.h, 484
- lbm\_hfx\_attr\_option\_size
  - lbm.h, 484
- lbm\_hfx\_attr\_set\_from\_xml
  - lbm.h, 484
- lbm\_hfx\_attr\_setopt
  - lbm.h, 484
- lbm\_hfx\_attr\_str\_getopt
  - lbm.h, 485
- lbm\_hfx\_attr\_str\_setopt
  - lbm.h, 485
- lbm\_hfx\_create
  - lbm.h, 486
- lbm\_hfx\_delete
  - lbm.h, 486
- lbm\_hfx\_delete\_ex
  - lbm.h, 486
- lbm\_hfx\_dump
  - lbm.h, 487
- lbm\_hfx\_getopt
  - lbm.h, 487
- lbm\_hfx\_rcv\_create
  - lbm.h, 488
- lbm\_hfx\_rcv\_delete
  - lbm.h, 488
- lbm\_hfx\_rcv\_delete\_ex
  - lbm.h, 488
- lbm\_hfx\_rcv\_topic\_dump
  - lbm.h, 489
- lbm\_hfx\_setopt
  - lbm.h, 489
- lbm\_hfx\_str\_getopt
  - lbm.h, 490
- lbm\_hfx\_str\_setopt
  - lbm.h, 490
- lbm\_hypertopic\_rcv\_add
  - lbmht.h, 565
- lbm\_hypertopic\_rcv\_cb\_proc

- lbmht.h, 564
- lbm\_hypertopic\_rcv\_delete
  - lbmht.h, 565
- lbm\_hypertopic\_rcv\_destroy
  - lbmht.h, 565
- lbm\_hypertopic\_rcv\_init
  - lbmht.h, 566
- lbm\_immediate\_msg\_cb\_proc
  - lbm.h, 413
- lbm\_iovec\_t
  - lbm.h, 413
- lbm\_iovec\_t\_stct, 148
  - iov\_base, 148
  - iov\_len, 148
- lbm\_ipv4\_address\_mask\_t
  - lbm.h, 413
- lbm\_ipv4\_address\_mask\_t\_stct, 149
  - addr, 149
  - bits, 149
- lbm\_is\_ume\_capable
  - lbm.h, 491
- lbm\_is\_umq\_capable
  - lbm.h, 491
- lbm\_license\_file
  - lbm.h, 491
- lbm\_license\_str
  - lbm.h, 491
- lbm\_license\_ummmn\_valid
  - lbm.h, 492
- lbm\_license\_vds\_valid
  - lbm.h, 492
- lbm\_log
  - lbm.h, 492
- LBM\_LOG\_ALERT
  - lbm.h, 383
- lbm\_log\_cb\_proc
  - lbm.h, 414
- LBM\_LOG\_CRIT
  - lbm.h, 383
- LBM\_LOG\_DEBUG
  - lbm.h, 383
- LBM\_LOG\_EMERG
  - lbm.h, 383
- LBM\_LOG\_ERR
  - lbm.h, 383
- LBM\_LOG\_INFO
  - lbm.h, 383
- LBM\_LOG\_NOTICE
  - lbm.h, 384
- LBM\_LOG\_WARNING
  - lbm.h, 384
- lbm\_logf
  - lbm.h, 492
- lbm\_mim\_unrecloss\_func\_t
  - lbm.h, 414
- lbm\_mim\_unrecloss\_func\_t\_stct, 150
- lbm\_mim\_unrecloss\_function\_cb
  - lbm.h, 414
- LBM\_MSG\_BOS
  - lbm.h, 384
- lbm\_msg\_channel\_info\_t
  - lbm.h, 415
- lbm\_msg\_channel\_info\_t\_stct, 151
  - channel\_number, 151
  - flags, 151
- LBM\_MSG\_COMPLETE\_BATCH
  - lbm.h, 384
- LBM\_MSG\_DATA
  - lbm.h, 384
- lbm\_msg\_delete
  - lbm.h, 492
- LBM\_MSG\_END\_BATCH
  - lbm.h, 384
- LBM\_MSG\_EOS
  - lbm.h, 384
- lbm\_msg\_extract\_ume\_ack
  - lbm.h, 493
- LBM\_MSG\_FLAG\_DELIVERY\_-  
LATENCY
  - lbm.h, 384
- LBM\_MSG\_FLAG\_END\_BATCH
  - lbm.h, 384
- LBM\_MSG\_FLAG\_HF\_32
  - lbm.h, 385
- LBM\_MSG\_FLAG\_HF\_64
  - lbm.h, 385
- LBM\_MSG\_FLAG\_HF\_DUPLICATE
  - lbm.h, 385
- LBM\_MSG\_FLAG\_HF\_OPTIONAL
  - lbm.h, 385
- LBM\_MSG\_FLAG\_HF\_PASS\_-  
THROUGH

- lbm.h, [385](#)
- LBM\_MSG\_FLAG\_IMMEDIATE
  - lbm.h, [385](#)
- LBM\_MSG\_FLAG\_NUMBERED\_-  
CHANNEL
  - lbm.h, [385](#)
- LBM\_MSG\_FLAG\_OTR
  - lbm.h, [385](#)
- LBM\_MSG\_FLAG\_RETRANSMIT
  - lbm.h, [385](#)
- LBM\_MSG\_FLAG\_START\_BATCH
  - lbm.h, [385](#)
- LBM\_MSG\_FLAG\_TOPICLESS
  - lbm.h, [386](#)
- LBM\_MSG\_FLAG\_UME\_-  
RETRANSMIT
  - lbm.h, [386](#)
- LBM\_MSG\_FLAG\_UME\_SRC\_REGID
  - lbm.h, [386](#)
- LBM\_MSG\_FLAG\_UMQ\_-  
REASSIGNED
  - lbm.h, [386](#)
- LBM\_MSG\_FLAG\_UMQ\_-  
RESUBMITTED
  - lbm.h, [386](#)
- LBM\_MSG\_FLUSH
  - lbm.h, [386](#)
- lbm\_msg\_fragment\_info\_t
  - lbm.h, [415](#)
- lbm\_msg\_fragment\_info\_t\_stct, [152](#)
  - offset, [152](#)
  - start\_sequence\_number, [152](#)
  - total\_message\_length, [152](#)
- lbm\_msg\_gateway\_info\_t
  - lbm.h, [415](#)
- lbm\_msg\_gateway\_info\_t\_stct, [153](#)
  - sequence\_number, [153](#)
  - source, [153](#)
- LBM\_MSG\_HF\_RESET
  - lbm.h, [386](#)
- LBM\_MSG\_IOV\_GATHER
  - lbm.h, [386](#)
- lbm\_msg\_is\_fragment
  - lbm.h, [493](#)
- LBM\_MSG\_NO\_SOURCE\_-  
NOTIFICATION
  - lbm.h, [387](#)
- lbm\_msg\_properties\_clear
  - lbm.h, [493](#)
- lbm\_msg\_properties\_create
  - lbm.h, [494](#)
- lbm\_msg\_properties\_delete
  - lbm.h, [494](#)
- lbm\_msg\_properties\_get
  - lbm.h, [495](#)
- lbm\_msg\_properties\_iter\_create
  - lbm.h, [495](#)
- lbm\_msg\_properties\_iter\_delete
  - lbm.h, [496](#)
- lbm\_msg\_properties\_iter\_first
  - lbm.h, [496](#)
- lbm\_msg\_properties\_iter\_next
  - lbm.h, [497](#)
- lbm\_msg\_properties\_iter\_t\_stct, [154](#)
- LBM\_MSG\_PROPERTIES\_MAX\_-  
NAMELEN
  - lbm.h, [387](#)
- lbm\_msg\_properties\_set
  - lbm.h, [497](#)
- LBM\_MSG\_PROPERTY\_BOOLEAN
  - lbm.h, [387](#)
- LBM\_MSG\_PROPERTY\_BYTE
  - lbm.h, [387](#)
- LBM\_MSG\_PROPERTY\_DOUBLE
  - lbm.h, [387](#)
- LBM\_MSG\_PROPERTY\_FLOAT
  - lbm.h, [387](#)
- LBM\_MSG\_PROPERTY\_INT
  - lbm.h, [387](#)
- LBM\_MSG\_PROPERTY\_LONG
  - lbm.h, [387](#)
- LBM\_MSG\_PROPERTY\_NONE
  - lbm.h, [387](#)
- LBM\_MSG\_PROPERTY\_SHORT
  - lbm.h, [388](#)
- LBM\_MSG\_PROPERTY\_STRING
  - lbm.h, [388](#)
- LBM\_MSG\_REQUEST
  - lbm.h, [388](#)
- LBM\_MSG\_RESPONSE
  - lbm.h, [388](#)
- lbm\_msg\_retain

- lbm.h, 498
- lbm\_msg\_retrieve\_fragment\_info
  - lbm.h, 498
- lbm\_msg\_retrieve\_gateway\_info
  - lbm.h, 498
- lbm\_msg\_retrieve\_msgid
  - lbm.h, 498
- lbm\_msg\_retrieve\_umq\_index
  - lbm.h, 499
- LBM\_MSG\_START\_BATCH
  - lbm.h, 388
- lbm\_msg\_t\_stct, 155
  - channel\_info, 156
  - copied\_state, 156
  - data, 156
  - flags, 156
  - hf\_sequence\_number, 156
  - len, 156
  - properties, 157
  - response, 157
  - sequence\_number, 157
  - source, 157
  - source\_clientd, 157
  - topic\_name, 157
  - type, 157
- lbm\_msg\_ume\_can\_send\_explicit\_ack
  - lbm.h, 499
- LBM\_MSG\_UME\_-
  - DEREGISTRATION\_-
  - COMPLETE\_EX
  - lbm.h, 388
- lbm\_msg\_ume\_deregistration\_ex\_t
  - lbm.h, 415
- lbm\_msg\_ume\_deregistration\_ex\_t\_stct, 160
  - flags, 160
  - rcv\_registration\_id, 160
  - sequence\_number, 160
  - src\_registration\_id, 160
  - store, 161
  - store\_index, 161
- LBM\_MSG\_UME\_-
  - DEREGISTRATION\_-
  - SUCCESS\_EX
  - lbm.h, 388
- LBM\_MSG\_UME\_-
  - DEREGISTRATION\_-
  - SUCCESS\_EX\_FLAG\_RPP
  - lbm.h, 388
- LBM\_MSG\_UME\_REGISTRATION\_-
  - CHANGE
  - lbm.h, 388
- LBM\_MSG\_UME\_REGISTRATION\_-
  - COMPLETE\_EX
  - lbm.h, 389
- LBM\_MSG\_UME\_REGISTRATION\_-
  - COMPLETE\_EX\_FLAG\_-
  - QUORUM
  - lbm.h, 389
- LBM\_MSG\_UME\_REGISTRATION\_-
  - COMPLETE\_EX\_FLAG\_-
  - RXREQMAX
  - lbm.h, 389
- LBM\_MSG\_UME\_REGISTRATION\_-
  - COMPLETE\_EX\_FLAG\_-
  - SRC\_SID
  - lbm.h, 389
- lbm\_msg\_ume\_registration\_complete\_-
  - ex\_t
  - lbm.h, 415
- lbm\_msg\_ume\_registration\_complete\_-
  - ex\_t\_stct, 162
  - flags, 162
  - sequence\_number, 162
  - src\_session\_id, 162
- LBM\_MSG\_UME\_REGISTRATION\_-
  - ERROR
  - lbm.h, 389
- lbm\_msg\_ume\_registration\_ex\_t
  - lbm.h, 416
- lbm\_msg\_ume\_registration\_ex\_t\_stct, 163
  - flags, 163
  - rcv\_registration\_id, 163
  - sequence\_number, 163
  - src\_registration\_id, 163
  - src\_session\_id, 163
  - store, 164
  - store\_index, 164
- LBM\_MSG\_UME\_REGISTRATION\_-
  - SUCCESS

- lbm.h, [389](#)
- LBM\_MSG\_UME\_REGISTRATION\_-  
SUCCESS\_EX
- lbm.h, [389](#)
- LBM\_MSG\_UME\_REGISTRATION\_-  
SUCCESS\_EX\_FLAG\_-  
NOCACHE
- lbm.h, [389](#)
- LBM\_MSG\_UME\_REGISTRATION\_-  
SUCCESS\_EX\_FLAG\_OLD
- lbm.h, [390](#)
- LBM\_MSG\_UME\_REGISTRATION\_-  
SUCCESS\_EX\_FLAG\_RPP
- lbm.h, [390](#)
- LBM\_MSG\_UME\_REGISTRATION\_-  
SUCCESS\_EX\_FLAG\_SRC\_-  
SID
- lbm.h, [390](#)
- lbm\_msg\_ume\_registration\_t
- lbm.h, [416](#)
- lbm\_msg\_ume\_registration\_t\_stct, [165](#)
- rcv\_registration\_id, [165](#)
- src\_registration\_id, [165](#)
- lbm\_msg\_ume\_send\_explicit\_ack
- lbm.h, [499](#)
- LBM\_MSG\_UMQ\_-  
DEREGISTRATION\_-  
COMPLETE\_EX
- lbm.h, [390](#)
- LBM\_MSG\_UMQ\_-  
DEREGISTRATION\_-  
COMPLETE\_EX\_FLAG\_ULB
- lbm.h, [390](#)
- lbm\_msg\_umq\_deregistration\_-  
complete\_ex\_t
- lbm.h, [416](#)
- lbm\_msg\_umq\_deregistration\_-  
complete\_ex\_t\_stct, [166](#)
- flags, [166](#)
- queue\_id, [166](#)
- LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNED\_EX
- lbm.h, [390](#)
- LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNED\_EX\_FLAG\_-  
REQUESTED
- lbm.h, [390](#)
- LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNED\_EX\_FLAG\_ULB
- lbm.h, [390](#)
- lbm\_msg\_umq\_index\_assigned\_ex\_t
- lbm.h, [416](#)
- lbm\_msg\_umq\_index\_assigned\_ex\_t\_-  
stct, [167](#)
- flags, [167](#)
- queue\_id, [167](#)
- LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNMENT\_-  
ELIGIBILITY\_ERROR
- lbm.h, [391](#)
- LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNMENT\_-  
ELIGIBILITY\_START\_-  
COMPLETE\_EX
- lbm.h, [391](#)
- LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNMENT\_-  
ELIGIBILITY\_START\_-  
COMPLETE\_EX\_FLAG\_ULB
- lbm.h, [391](#)
- lbm\_msg\_umq\_index\_assignment\_-  
eligibility\_start\_complete\_ex\_t
- lbm.h, [416](#)
- lbm\_msg\_umq\_index\_assignment\_-  
eligibility\_start\_complete\_ex\_-  
t\_stct, [168](#)
- flags, [168](#)
- queue\_id, [168](#)
- LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNMENT\_-  
ELIGIBILITY\_STOP\_-  
COMPLETE\_EX
- lbm.h, [391](#)
- LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNMENT\_-  
ELIGIBILITY\_STOP\_-  
COMPLETE\_EX\_FLAG\_ULB
- lbm.h, [391](#)
- lbm\_msg\_umq\_index\_assignment\_-  
eligibility\_stop\_complete\_ex\_t
- lbm.h, [416](#)

- lbm\_msg\_umq\_index\_assignment\_-  
eligibility\_stop\_complete\_ex\_-  
t\_stct, [169](#)  
flags, [169](#)  
queue\_id, [169](#)
- LBM\_MSG\_UMQ\_INDEX\_-  
ASSIGNMENT\_ERROR  
lbm.h, [391](#)
- LBM\_MSG\_UMQ\_INDEX\_-  
RELEASED\_EX  
lbm.h, [391](#)
- LBM\_MSG\_UMQ\_INDEX\_-  
RELEASED\_EX\_FLAG\_ULB  
lbm.h, [391](#)
- lbm\_msg\_umq\_index\_released\_ex\_t  
lbm.h, [416](#)
- lbm\_msg\_umq\_index\_released\_ex\_t\_-  
stct, [170](#)  
flags, [170](#)  
queue\_id, [170](#)
- lbm\_msg\_umq\_reassign  
lbm.h, [500](#)
- LBM\_MSG\_UMQ\_REASSIGN\_-  
FLAG\_DISCARD  
lbm.h, [392](#)
- LBM\_MSG\_UMQ\_REGISTRATION\_-  
COMPLETE\_EX  
lbm.h, [392](#)
- LBM\_MSG\_UMQ\_REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_-  
QUORUM  
lbm.h, [392](#)
- LBM\_MSG\_UMQ\_REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_ULB  
lbm.h, [392](#)
- lbm\_msg\_umq\_registration\_complete\_-  
ex\_t  
lbm.h, [417](#)
- lbm\_msg\_umq\_registration\_complete\_-  
ex\_t\_stct, [171](#)  
assignment\_id, [171](#)  
flags, [171](#)  
queue, [171](#)  
queue\_id, [171](#)
- LBM\_MSG\_UMQ\_REGISTRATION\_-  
ERROR  
lbm.h, [392](#)
- LBM\_MSG\_UNRECOVERABLE\_-  
LOSS  
lbm.h, [392](#)
- LBM\_MSG\_UNRECOVERABLE\_-  
LOSS\_BURST  
lbm.h, [392](#)
- lbm\_msgs\_no\_topic\_rcved
- lbm\_rcv\_transport\_stats\_lbtpc\_t\_-  
stct, [177](#)
- lbm\_rcv\_transport\_stats\_lbrdma\_-  
t\_stct, [179](#)
- lbm\_rcv\_transport\_stats\_lbrm\_t\_-  
stct, [182](#)
- lbm\_rcv\_transport\_stats\_lbrtru\_t\_-  
stct, [190](#)
- lbm\_rcv\_transport\_stats\_lbtmsx\_t\_-  
stct, [194](#)
- lbm\_rcv\_transport\_stats\_tcp\_t\_stct,  
[199](#)
- lbm\_msgs\_rcved
- lbm\_rcv\_transport\_stats\_lbtpc\_t\_-  
stct, [177](#)
- lbm\_rcv\_transport\_stats\_lbrdma\_-  
t\_stct, [179](#)
- lbm\_rcv\_transport\_stats\_lbrm\_t\_-  
stct, [183](#)
- lbm\_rcv\_transport\_stats\_lbrtru\_t\_-  
stct, [190](#)
- lbm\_rcv\_transport\_stats\_lbtmsx\_t\_-  
stct, [194](#)
- lbm\_rcv\_transport\_stats\_tcp\_t\_stct,  
[199](#)
- lbm\_multicast\_immediate\_message  
lbm.h, [500](#)
- lbm\_multicast\_immediate\_request  
lbm.h, [500](#)
- lbm\_queue\_immediate\_message  
lbm.h, [501](#)
- LBM\_RCV\_BLOCK  
lbm.h, [392](#)
- LBM\_RCV\_BLOCK\_TEMP  
lbm.h, [392](#)
- lbm\_rcv\_cb\_proc  
lbm.h, [417](#)
- lbm\_rcv\_create

- lbm.h, [502](#)
- lbm\_rcv\_delete
  - lbm.h, [502](#)
- lbm\_rcv\_delete\_ex
  - lbm.h, [503](#)
- lbm\_rcv\_from\_hf\_rcv
  - lbm.h, [503](#)
- lbm\_rcv\_from\_hfx\_rcv
  - lbm.h, [504](#)
- lbm\_rcv\_getopt
  - lbm.h, [504](#)
- lbm\_rcv\_msg\_source\_clientd
  - lbm.h, [504](#)
- LBM\_RCV\_NONBLOCK
  - lbm.h, [393](#)
- lbm\_rcv\_reset\_all\_transport\_stats
  - lbm.h, [504](#)
- lbm\_rcv\_reset\_transport\_stats
  - lbm.h, [505](#)
- lbm\_rcv\_retrieve\_all\_transport\_stats
  - lbm.h, [505](#)
- lbm\_rcv\_retrieve\_transport\_stats
  - lbm.h, [505](#)
- lbm\_rcv\_setopt
  - lbm.h, [506](#)
- lbm\_rcv\_src\_notification\_create\_ -
  - function\_cb
  - lbm.h, [417](#)
- lbm\_rcv\_src\_notification\_delete\_ -
  - function\_cb
  - lbm.h, [418](#)
- lbm\_rcv\_src\_notification\_func\_t
  - lbm.h, [418](#)
- lbm\_rcv\_src\_notification\_func\_t\_stct,
  - [173](#)
- lbm\_rcv\_str\_getopt
  - lbm.h, [506](#)
- lbm\_rcv\_str\_setopt
  - lbm.h, [507](#)
- lbm\_rcv\_subscribe\_channel
  - lbm.h, [507](#)
- lbm\_rcv\_topic\_attr\_create
  - lbm.h, [508](#)
- lbm\_rcv\_topic\_attr\_create\_default
  - lbm.h, [508](#)
- lbm\_rcv\_topic\_attr\_create\_from\_xml
  - lbm.h, [508](#)
- lbm\_rcv\_topic\_attr\_delete
  - lbm.h, [509](#)
- lbm\_rcv\_topic\_attr\_dump
  - lbm.h, [509](#)
- lbm\_rcv\_topic\_attr\_dup
  - lbm.h, [509](#)
- lbm\_rcv\_topic\_attr\_getopt
  - lbm.h, [510](#)
- lbm\_rcv\_topic\_attr\_option\_size
  - lbm.h, [510](#)
- lbm\_rcv\_topic\_attr\_set\_from\_xml
  - lbm.h, [510](#)
- lbm\_rcv\_topic\_attr\_setopt
  - lbm.h, [511](#)
- lbm\_rcv\_topic\_attr\_str\_getopt
  - lbm.h, [511](#)
- lbm\_rcv\_topic\_attr\_str\_setopt
  - lbm.h, [512](#)
- lbm\_rcv\_topic\_dump
  - lbm.h, [512](#)
- lbm\_rcv\_topic\_lookup
  - lbm.h, [512](#)
- LBM\_RCV\_TOPIC\_STATS\_FLAG\_ -
  - SRC\_VALID
  - lbm.h, [393](#)
- lbm\_rcv\_topic\_stats\_t
  - lbm.h, [418](#)
- lbm\_rcv\_topic\_stats\_t\_stct, [174](#)
  - flags, [174](#)
  - otid, [174](#)
  - source, [174](#)
  - topic, [174](#)
  - topic\_idx, [174](#)
- lbm\_rcv\_transport\_stats\_daemon\_t
  - lbm.h, [418](#)
- lbm\_rcv\_transport\_stats\_daemon\_t\_stct,
  - [176](#)
  - bytes\_rcved, [176](#)
- lbm\_rcv\_transport\_stats\_lbtipc\_t\_stct,
  - [177](#)
  - bytes\_rcved, [177](#)
  - lbm\_msgs\_no\_topic\_rcved, [177](#)
  - lbm\_msgs\_rcved, [177](#)
  - lbm\_reqs\_rcved, [177](#)
  - msgs\_rcved, [178](#)

- lbm\_rcv\_transport\_stats\_lbtrdma\_t\_stct, 179
  - bytes\_rcved, 179
  - lbm\_msgs\_no\_topic\_rcved, 179
  - lbm\_msgs\_rcved, 179
  - lbm\_reqs\_rcved, 179
  - msgs\_rcved, 180
- lbm\_rcv\_transport\_stats\_lbtrm\_t\_stct, 181
  - bytes\_rcved, 181
  - dgrams\_dropped\_hdr, 181
  - dgrams\_dropped\_other, 182
  - dgrams\_dropped\_size, 182
  - dgrams\_dropped\_type, 182
  - dgrams\_dropped\_version, 182
  - duplicate\_data, 182
  - lbm\_msgs\_no\_topic\_rcved, 182
  - lbm\_msgs\_rcved, 183
  - lbm\_reqs\_rcved, 183
  - lost, 183
  - msgs\_rcved, 183
  - nak\_pckts\_sent, 183
  - nak\_stm\_max, 184
  - nak\_stm\_mean, 184
  - nak\_stm\_min, 184
  - nak\_tx\_max, 184
  - nak\_tx\_mean, 184
  - nak\_tx\_min, 184
  - naks\_sent, 185
  - ncfs\_ignored, 185
  - ncfs\_rx\_delay, 185
  - ncfs\_shed, 185
  - ncfs\_unknown, 186
  - out\_of\_order, 186
  - unrecovered\_tmo, 186
  - unrecovered\_twx, 186
- lbm\_rcv\_transport\_stats\_lbtru\_t\_stct, 188
  - bytes\_rcved, 188
  - dgrams\_dropped\_hdr, 188
  - dgrams\_dropped\_other, 189
  - dgrams\_dropped\_sid, 189
  - dgrams\_dropped\_size, 189
  - dgrams\_dropped\_type, 189
  - dgrams\_dropped\_version, 189
  - duplicate\_data, 189
  - lbm\_msgs\_no\_topic\_rcved, 190
- lbm\_msgs\_rcved, 190
- lbm\_reqs\_rcved, 190
- lost, 190
- msgs\_rcved, 190
- nak\_pckts\_sent, 190
- nak\_stm\_max, 190
- nak\_stm\_mean, 191
- nak\_stm\_min, 191
- nak\_tx\_max, 191
- nak\_tx\_mean, 191
- nak\_tx\_min, 191
- naks\_sent, 191
- ncfs\_ignored, 192
- ncfs\_rx\_delay, 192
- ncfs\_shed, 192
- ncfs\_unknown, 192
- unrecovered\_tmo, 193
- unrecovered\_twx, 193
- lbm\_rcv\_transport\_stats\_lbtstmx\_t\_stct, 194
  - bytes\_rcved, 194
  - lbm\_msgs\_no\_topic\_rcved, 194
  - lbm\_msgs\_rcved, 194
  - msgs\_rcved, 194
  - reserved1, 194
- lbm\_rcv\_transport\_stats\_t
  - lbm.h, 419
- lbm\_rcv\_transport\_stats\_t\_stct, 196
  - daemon, 197
  - lbtipc, 197
  - lbtrdma, 197
  - lbtrm, 197
  - lbtru, 197
  - lbtstmx, 197
  - source, 197
  - tcp, 197
  - type, 197
- lbm\_rcv\_transport\_stats\_tcp\_t\_stct, 199
  - bytes\_rcved, 199
  - lbm\_msgs\_no\_topic\_rcved, 199
  - lbm\_msgs\_rcved, 199
  - lbm\_reqs\_rcved, 199
- lbm\_rcv\_ume\_deregister
  - lbm.h, 513
- lbm\_rcv\_umq\_deregister
  - lbm.h, 513

- lbm\_rcv\_umq\_index\_release  
lbm.h, [513](#)
- lbm\_rcv\_umq\_index\_reserve  
lbm.h, [514](#)
- lbm\_rcv\_umq\_index\_start\_assignment  
lbm.h, [514](#)
- lbm\_rcv\_umq\_index\_stop\_assignment  
lbm.h, [514](#)
- lbm\_rcv\_umq\_queue\_msg\_list  
lbm.h, [515](#)
- lbm\_rcv\_umq\_queue\_msg\_list\_info\_t,  
[201](#)  
msgs, [201](#)  
num\_msgs, [201](#)
- lbm\_rcv\_umq\_queue\_msg\_retrieve  
lbm.h, [515](#)
- lbm\_rcv\_umq\_queue\_msg\_retrieve\_  
info\_t, [202](#)  
msgs, [202](#)  
num\_msgs, [202](#)
- lbm\_rcv\_unsubscribe\_channel  
lbm.h, [516](#)
- lbm\_rcv\_unsubscribe\_channel\_ex  
lbm.h, [517](#)
- lbm\_register\_fd  
lbm.h, [517](#)
- lbm\_reqs\_rcved  
lbm\_rcv\_transport\_stats\_lbtipc\_t\_  
stct, [177](#)  
lbm\_rcv\_transport\_stats\_lbtrdma\_  
t\_stct, [179](#)  
lbm\_rcv\_transport\_stats\_lbtrm\_t\_  
stct, [183](#)  
lbm\_rcv\_transport\_stats\_lbtru\_t\_  
stct, [190](#)  
lbm\_rcv\_transport\_stats\_tcp\_t\_stct,  
[199](#)
- lbm\_request\_cb\_proc  
lbm.h, [419](#)
- lbm\_request\_delete  
lbm.h, [518](#)
- lbm\_request\_delete\_ex  
lbm.h, [518](#)
- lbm\_resolver\_event\_advertisement\_t\_  
stct, [203](#)
- lbm\_resolver\_event\_func\_t\_stct, [204](#)
- lbm\_resolver\_event\_info\_t\_stct, [205](#)
- lbm\_response\_delete  
lbm.h, [518](#)
- lbm\_schedule\_timer  
lbm.h, [519](#)
- lbm\_schedule\_timer\_recurring  
lbm.h, [520](#)
- lbm\_send\_request  
lbm.h, [520](#)
- lbm\_send\_request\_ex  
lbm.h, [521](#)
- lbm\_send\_response  
lbm.h, [522](#)
- lbm\_serialize\_response  
lbm.h, [522](#)
- lbm\_serialized\_response\_delete  
lbm.h, [523](#)
- lbm\_serialized\_response\_t\_stct, [206](#)
- lbm\_set\_lbtrm\_loss\_rate  
lbm.h, [523](#)
- lbm\_set\_lbtrm\_src\_loss\_rate  
lbm.h, [523](#)
- lbm\_set\_lbtru\_loss\_rate  
lbm.h, [523](#)
- lbm\_set\_lbtru\_src\_loss\_rate  
lbm.h, [524](#)
- lbm\_set\_uim\_loss\_rate  
lbm.h, [524](#)
- lbm\_set\_umm\_info  
lbm.h, [524](#)
- LBM\_SRC\_BLOCK  
lbm.h, [393](#)
- LBM\_SRC\_BLOCK\_TEMP  
lbm.h, [393](#)
- lbm\_src\_buff\_acquire  
lbm.h, [524](#)
- lbm\_src\_buffs\_cancel  
lbm.h, [525](#)
- lbm\_src\_buffs\_complete  
lbm.h, [525](#)
- lbm\_src\_buffs\_complete\_and\_acquire  
lbm.h, [526](#)
- lbm\_src\_cb\_proc  
lbm.h, [419](#)
- lbm\_src\_channel\_create  
lbm.h, [526](#)

- lbm\_src\_channel\_delete
  - lbm.h, [526](#)
- lbm\_src\_cost\_func\_t\_stct, [207](#)
- lbm\_src\_cost\_function\_cb
  - lbm.h, [421](#)
- LBM\_SRC\_COST\_FUNCTION\_-REJECT
  - lbm.h, [393](#)
- lbm\_src\_create
  - lbm.h, [527](#)
- lbm\_src\_delete
  - lbm.h, [527](#)
- lbm\_src\_delete\_ex
  - lbm.h, [528](#)
- LBM\_SRC\_EVENT\_CONNECT
  - lbm.h, [393](#)
- LBM\_SRC\_EVENT\_DAEMON\_-CONFIRM
  - lbm.h, [393](#)
- LBM\_SRC\_EVENT\_DISCONNECT
  - lbm.h, [394](#)
- LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_-NOTIFICATION
  - lbm.h, [394](#)
- LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_-NOTIFICATION\_STATE\_-OVER
  - lbm.h, [394](#)
- LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_-NOTIFICATION\_STATE\_-UNDER
  - lbm.h, [394](#)
- lbm\_src\_event\_flight\_size\_notification\_t
  - lbm.h, [421](#)
- lbm\_src\_event\_flight\_size\_notification\_-t\_stct, [208](#)
  - state, [208](#)
  - type, [208](#)
- LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_-NOTIFICATION\_TYPE\_ULB
  - lbm.h, [394](#)
- LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_-NOTIFICATION\_TYPE\_UME
  - lbm.h, [394](#)
- LBM\_SRC\_EVENT\_FLIGHT\_SIZE\_-NOTIFICATION\_TYPE\_UMQ
  - lbm.h, [394](#)
- LBM\_SRC\_EVENT\_SEQUENCE\_-NUMBER\_INFO
  - lbm.h, [395](#)
- lbm\_src\_event\_sequence\_number\_info\_t
  - lbm.h, [421](#)
- lbm\_src\_event\_sequence\_number\_info\_-t\_stct, [209](#)
  - first\_sequence\_number, [209](#)
  - flags, [209](#)
  - last\_sequence\_number, [209](#)
  - msg\_clientd, [209](#)
- lbm\_src\_event\_ume\_ack\_ex\_info\_t
  - lbm.h, [422](#)
- lbm\_src\_event\_ume\_ack\_ex\_info\_t\_stct, [211](#)
  - flags, [211](#)
  - msg\_clientd, [211](#)
  - rcv\_registration\_id, [211](#)
  - sequence\_number, [211](#)
  - store, [211](#)
  - store\_index, [212](#)
- lbm\_src\_event\_ume\_ack\_info\_t
  - lbm.h, [422](#)
- lbm\_src\_event\_ume\_ack\_info\_t\_stct, [213](#)
  - msg\_clientd, [213](#)
  - rcv\_registration\_id, [213](#)
  - sequence\_number, [213](#)
- LBM\_SRC\_EVENT\_UME\_-DELIVERY\_-CONFIRMATION
  - lbm.h, [395](#)
- LBM\_SRC\_EVENT\_UME\_-DELIVERY\_-CONFIRMATION\_EX
  - lbm.h, [395](#)
- LBM\_SRC\_EVENT\_UME\_-DELIVERY\_-CONFIRMATION\_EX\_-FLAG\_EXACK
  - lbm.h, [395](#)
- LBM\_SRC\_EVENT\_UME\_-DELIVERY\_-CONFIRMATION\_EX\_-FLAG\_OOD
  - lbm.h, [395](#)

- LBM\_SRC\_EVENT\_UME\_-  
  - DELIVERY\_-
  - CONFIRMATION\_EX\_-
  - FLAG\_UNIQUEACKS
 lbm.h, [395](#)
- LBM\_SRC\_EVENT\_UME\_-  
  - DELIVERY\_-
  - CONFIRMATION\_EX\_-
  - FLAG\_UREGID
 lbm.h, [395](#)
- LBM\_SRC\_EVENT\_UME\_-  
  - DELIVERY\_-
  - CONFIRMATION\_EX\_-
  - FLAG\_WHOLE\_MESSAGE\_-
  - CONFIRMED
 lbm.h, [396](#)
- LBM\_SRC\_EVENT\_UME\_-  
  - DEREGISTRATION\_-
  - COMPLETE\_EX
 lbm.h, [396](#)
- lbm\_src\_event\_ume\_deregistration\_ex\_t  
 lbm.h, [422](#)
- lbm\_src\_event\_ume\_deregistration\_ex\_-  
  - t\_stct, [214](#)
  - flags, [214](#)
  - registration\_id, [214](#)
  - sequence\_number, [214](#)
  - store, [214](#)
  - store\_index, [215](#)
- LBM\_SRC\_EVENT\_UME\_-  
  - DEREGISTRATION\_-
  - SUCCESS\_EX
 lbm.h, [396](#)
- LBM\_SRC\_EVENT\_UME\_-  
  - MESSAGE\_NOT\_STABLE
 lbm.h, [396](#)
- LBM\_SRC\_EVENT\_UME\_-  
  - MESSAGE\_NOT\_STABLE\_-
  - FLAG\_LOSS
 lbm.h, [396](#)
- LBM\_SRC\_EVENT\_UME\_-  
  - MESSAGE\_NOT\_STABLE\_-
  - FLAG\_STORE
 lbm.h, [396](#)
- LBM\_SRC\_EVENT\_UME\_-  
  - MESSAGE\_NOT\_STABLE\_-
- LBM\_SRC\_EVENT\_UME\_-  
  - FLAG\_TIMEOUT
 lbm.h, [396](#)
- LBM\_SRC\_EVENT\_UME\_-  
  - MESSAGE\_RECLAIMED
 lbm.h, [397](#)
- LBM\_SRC\_EVENT\_UME\_-  
  - MESSAGE\_RECLAIMED\_-
  - EX
 lbm.h, [397](#)
- LBM\_SRC\_EVENT\_UME\_-  
  - MESSAGE\_RECLAIMED\_-
  - EX\_FLAG\_FORCED
 lbm.h, [397](#)
- LBM\_SRC\_EVENT\_UME\_-  
  - MESSAGE\_STABLE
 lbm.h, [397](#)
- LBM\_SRC\_EVENT\_UME\_-  
  - MESSAGE\_STABLE\_EX
 lbm.h, [397](#)
- LBM\_SRC\_EVENT\_UME\_-  
  - MESSAGE\_STABLE\_EX\_-
  - FLAG\_INTERGROUP\_-
  - STABLE
 lbm.h, [397](#)
- LBM\_SRC\_EVENT\_UME\_-  
  - MESSAGE\_STABLE\_EX\_-
  - FLAG\_INTRAGROUP\_-
  - STABLE
 lbm.h, [397](#)
- LBM\_SRC\_EVENT\_UME\_-  
  - MESSAGE\_STABLE\_EX\_-
  - FLAG\_STABLE
 lbm.h, [397](#)
- LBM\_SRC\_EVENT\_UME\_-  
  - MESSAGE\_STABLE\_EX\_-
  - FLAG\_STORE
 lbm.h, [398](#)
- LBM\_SRC\_EVENT\_UME\_-  
  - MESSAGE\_STABLE\_EX\_-
  - FLAG\_USER
 lbm.h, [398](#)
- LBM\_SRC\_EVENT\_UME\_-  
  - MESSAGE\_STABLE\_EX\_-
  - FLAG\_WHOLE\_MESSAGE\_-
  - STABLE
 lbm.h, [398](#)

- LBM\_SRC\_EVENT\_UME\_-  
REGISTRATION\_-  
COMPLETE\_EX  
lbm.h, [398](#)
- LBM\_SRC\_EVENT\_UME\_-  
REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_-  
QUORUM  
lbm.h, [398](#)
- lbm\_src\_event\_ume\_registration\_-  
complete\_ex\_t  
lbm.h, [422](#)
- lbm\_src\_event\_ume\_registration\_-  
complete\_ex\_t\_stct, [216](#)  
flags, [216](#)  
sequence\_number, [216](#)
- LBM\_SRC\_EVENT\_UME\_-  
REGISTRATION\_ERROR  
lbm.h, [398](#)
- lbm\_src\_event\_ume\_registration\_ex\_t  
lbm.h, [422](#)
- lbm\_src\_event\_ume\_registration\_ex\_t\_-  
stct, [217](#)  
flags, [217](#)  
registration\_id, [217](#)  
sequence\_number, [217](#)  
store, [217](#)  
store\_index, [217](#)
- LBM\_SRC\_EVENT\_UME\_-  
REGISTRATION\_SUCCESS  
lbm.h, [398](#)
- LBM\_SRC\_EVENT\_UME\_-  
REGISTRATION\_-  
SUCCESS\_EX  
lbm.h, [398](#)
- LBM\_SRC\_EVENT\_UME\_-  
REGISTRATION\_-  
SUCCESS\_EX\_FLAG\_-  
NOACKS  
lbm.h, [399](#)
- LBM\_SRC\_EVENT\_UME\_-  
REGISTRATION\_-  
SUCCESS\_EX\_FLAG\_OLD  
lbm.h, [399](#)
- LBM\_SRC\_EVENT\_UME\_-  
REGISTRATION\_-  
SUCCESS\_EX\_FLAG\_RPP  
lbm.h, [399](#)
- lbm\_src\_event\_ume\_registration\_t  
lbm.h, [422](#)
- lbm\_src\_event\_ume\_registration\_t\_stct,  
[219](#)  
registration\_id, [219](#)
- LBM\_SRC\_EVENT\_UME\_STORE\_-  
UNRESPONSIVE  
lbm.h, [399](#)
- LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_ID\_INFO  
lbm.h, [399](#)
- lbm\_src\_event\_umq\_message\_id\_info\_t  
lbm.h, [423](#)
- lbm\_src\_event\_umq\_message\_id\_info\_-  
t\_stct, [220](#)  
flags, [220](#)  
msg\_clientd, [220](#)  
msg\_id, [220](#)
- LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_STABLE\_EX  
lbm.h, [399](#)
- LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_INTERGROUP\_-  
STABLE  
lbm.h, [399](#)
- LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_INTRAGROUP\_-  
STABLE  
lbm.h, [400](#)
- LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_STABLE  
lbm.h, [400](#)
- LBM\_SRC\_EVENT\_UMQ\_-  
MESSAGE\_STABLE\_EX\_-  
FLAG\_USER  
lbm.h, [400](#)
- LBM\_SRC\_EVENT\_UMQ\_-  
REGISTRATION\_-  
COMPLETE\_EX  
lbm.h, [400](#)

- LBM\_SRC\_EVENT\_UMQ\_-  
REGISTRATION\_-  
COMPLETE\_EX\_FLAG\_-  
QUORUM  
lbm.h, 400
- lbm\_src\_event\_umq\_registration\_-  
complete\_ex\_t  
lbm.h, 423
- lbm\_src\_event\_umq\_registration\_-  
complete\_ex\_t\_stct, 222  
flags, 222  
queue, 222  
queue\_id, 222
- LBM\_SRC\_EVENT\_UMQ\_-  
REGISTRATION\_ERROR  
lbm.h, 400
- lbm\_src\_event\_umq\_stability\_ack\_info\_-  
ex\_t  
lbm.h, 423
- lbm\_src\_event\_umq\_stability\_ack\_info\_-  
ex\_t\_stct, 223  
first\_sequence\_number, 223  
flags, 223  
last\_sequence\_number, 223  
msg\_clientd, 224  
msg\_id, 224  
queue, 224  
queue\_id, 224  
queue\_instance, 224  
queue\_instance\_index, 224
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_ASSIGNED\_EX  
lbm.h, 400
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_COMPLETE\_EX  
lbm.h, 400
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_CONSUMED\_EX  
lbm.h, 401
- lbm\_src\_event\_umq\_ulb\_message\_info\_-  
ex\_t  
lbm.h, 423
- lbm\_src\_event\_umq\_ulb\_message\_info\_-  
ex\_t\_stct, 225  
application\_set\_index, 225  
assignment\_id, 225
- first\_sequence\_number, 225  
flags, 226  
last\_sequence\_number, 226  
msg\_clientd, 226  
msg\_id, 226  
receiver, 226  
registration\_id, 226
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_REASSIGNED\_-  
EX  
lbm.h, 401
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_REASSIGNED\_-  
EX\_FLAG\_EXPLICIT  
lbm.h, 401
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_TIMEOUT\_EX  
lbm.h, 401
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_TIMEOUT\_EX\_-  
FLAG\_DISCARD  
lbm.h, 401
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_TIMEOUT\_EX\_-  
FLAG\_EXPLICIT  
lbm.h, 401
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_TIMEOUT\_EX\_-  
FLAG\_MAX\_REASSIGNS  
lbm.h, 401
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
MESSAGE\_TIMEOUT\_EX\_-  
FLAG\_TOTAL\_LIFETIME\_-  
EXPIRED  
lbm.h, 402
- LBM\_SRC\_EVENT\_UMQ\_-  
ULB\_RECEIVER\_-  
DEREGISTRATION\_EX  
lbm.h, 402
- lbm\_src\_event\_umq\_ulb\_receiver\_info\_-  
ex\_t  
lbm.h, 423
- lbm\_src\_event\_umq\_ulb\_receiver\_info\_-  
ex\_t\_stct, 227  
application\_set\_index, 227  
assignment\_id, 227

- flags, [227](#)
- receiver, [227](#)
- registration\_id, [227](#)
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
RECEIVER\_READY\_EX  
lbm.h, [402](#)
- LBM\_SRC\_EVENT\_UMQ\_-  
ULB\_RECEIVER\_-  
REGISTRATION\_EX  
lbm.h, [402](#)
- LBM\_SRC\_EVENT\_UMQ\_ULB\_-  
RECEIVER\_TIMEOUT\_EX  
lbm.h, [402](#)
- LBM\_SRC\_EVENT\_WAKEUP  
lbm.h, [402](#)
- LBM\_SRC\_EVENT\_WAKEUP\_-  
FLAG\_MIM  
lbm.h, [402](#)
- LBM\_SRC\_EVENT\_WAKEUP\_-  
FLAG\_NORMAL  
lbm.h, [403](#)
- LBM\_SRC\_EVENT\_WAKEUP\_-  
FLAG\_REQUEST  
lbm.h, [403](#)
- LBM\_SRC\_EVENT\_WAKEUP\_-  
FLAG\_RESPONSE  
lbm.h, [403](#)
- LBM\_SRC\_EVENT\_WAKEUP\_-  
FLAG\_UIM  
lbm.h, [403](#)
- lbm\_src\_event\_wakeup\_t  
lbm.h, [423](#)
- lbm\_src\_event\_wakeup\_t\_stct, [229](#)  
flags, [229](#)
- lbm\_src\_flush  
lbm.h, [528](#)
- lbm\_src\_get\_inflight  
lbm.h, [528](#)
- lbm\_src\_get\_inflight\_ex  
lbm.h, [529](#)
- lbm\_src\_getopt  
lbm.h, [529](#)
- LBM\_SRC\_NONBLOCK  
lbm.h, [403](#)
- lbm\_src\_notify\_func\_t  
lbm.h, [423](#)
- lbm\_src\_notify\_func\_t\_stct, [230](#)
- lbm\_src\_notify\_function\_cb  
lbm.h, [424](#)
- lbm\_src\_reset\_transport\_stats  
lbm.h, [530](#)
- lbm\_src\_retrieve\_transport\_stats  
lbm.h, [530](#)
- lbm\_src\_send  
lbm.h, [530](#)
- lbm\_src\_send\_ex  
lbm.h, [531](#)
- LBM\_SRC\_SEND\_EX\_FLAG\_-  
APPHDR\_CHAIN  
lbm.h, [403](#)
- LBM\_SRC\_SEND\_EX\_FLAG\_-  
CHANNEL  
lbm.h, [403](#)
- LBM\_SRC\_SEND\_EX\_FLAG\_HF\_32  
lbm.h, [403](#)
- LBM\_SRC\_SEND\_EX\_FLAG\_HF\_64  
lbm.h, [403](#)
- LBM\_SRC\_SEND\_EX\_FLAG\_HF\_-  
OPTIONAL  
lbm.h, [404](#)
- LBM\_SRC\_SEND\_EX\_FLAG\_-  
PROPERTIES  
lbm.h, [404](#)
- LBM\_SRC\_SEND\_EX\_FLAG\_-  
SEQUENCE\_NUMBER\_-  
INFO  
lbm.h, [404](#)
- LBM\_SRC\_SEND\_EX\_FLAG\_-  
SEQUENCE\_NUMBER\_-  
INFO\_FRAGONLY  
lbm.h, [404](#)
- LBM\_SRC\_SEND\_EX\_FLAG\_UME\_-  
CLIENTD  
lbm.h, [404](#)
- LBM\_SRC\_SEND\_EX\_FLAG\_UMQ\_-  
CLIENTD  
lbm.h, [404](#)
- LBM\_SRC\_SEND\_EX\_FLAG\_UMQ\_-  
INDEX  
lbm.h, [404](#)
- LBM\_SRC\_SEND\_EX\_FLAG\_UMQ\_-  
MESSAGE\_ID\_INFO

- lbm.h, [404](#)
- LBM\_SRC\_SEND\_EX\_FLAG\_UMQ\_-  
TOTAL\_LIFETIME
  - lbm.h, [404](#)
- lbm\_src\_send\_ex\_info\_t
  - lbm.h, [424](#)
- lbm\_src\_send\_ex\_info\_t\_stct, [231](#)
  - apphdr\_chain, [231](#)
  - async\_opfunc, [231](#)
  - channel\_info, [231](#)
  - flags, [232](#)
  - hf\_sqn, [232](#)
  - properties, [232](#)
  - ume\_msg\_clientd, [232](#)
  - umq\_index, [232](#)
  - umq\_total\_lifetime, [232](#)
- lbm\_src\_sendv
  - lbm.h, [532](#)
- lbm\_src\_sendv\_ex
  - lbm.h, [533](#)
- lbm\_src\_setopt
  - lbm.h, [534](#)
- lbm\_src\_str\_getopt
  - lbm.h, [534](#)
- lbm\_src\_str\_setopt
  - lbm.h, [534](#)
- lbm\_src\_topic\_alloc
  - lbm.h, [535](#)
- lbm\_src\_topic\_attr\_create
  - lbm.h, [535](#)
- lbm\_src\_topic\_attr\_create\_default
  - lbm.h, [536](#)
- lbm\_src\_topic\_attr\_create\_from\_xml
  - lbm.h, [536](#)
- lbm\_src\_topic\_attr\_delete
  - lbm.h, [537](#)
- lbm\_src\_topic\_attr\_dump
  - lbm.h, [537](#)
- lbm\_src\_topic\_attr\_dup
  - lbm.h, [537](#)
- lbm\_src\_topic\_attr\_getopt
  - lbm.h, [537](#)
- lbm\_src\_topic\_attr\_option\_size
  - lbm.h, [538](#)
- lbm\_src\_topic\_attr\_set\_from\_xml
  - lbm.h, [538](#)
- lbm\_src\_topic\_attr\_setopt
  - lbm.h, [538](#)
- lbm\_src\_topic\_attr\_str\_getopt
  - lbm.h, [539](#)
- lbm\_src\_topic\_attr\_str\_setopt
  - lbm.h, [539](#)
- lbm\_src\_topic\_dump
  - lbm.h, [540](#)
- lbm\_src\_transport\_stats\_daemon\_t
  - lbm.h, [424](#)
- lbm\_src\_transport\_stats\_daemon\_t\_stct,  
[233](#)
  - bytes\_buffered, [233](#)
- lbm\_src\_transport\_stats\_lbtpc\_t\_stct,  
[234](#)
  - bytes\_sent, [234](#)
  - msgs\_sent, [234](#)
  - num\_clients, [234](#)
- lbm\_src\_transport\_stats\_lbtrdma\_t\_stct,  
[235](#)
  - bytes\_sent, [235](#)
  - msgs\_sent, [235](#)
  - num\_clients, [235](#)
- lbm\_src\_transport\_stats\_lbtrm\_t\_stct,  
[236](#)
  - bytes\_sent, [236](#)
  - msgs\_sent, [236](#)
  - nak\_pkts\_rcved, [236](#)
  - naks\_ignored, [236](#)
  - naks\_rcved, [237](#)
  - naks\_rx\_delay\_ignored, [237](#)
  - naks\_shed, [237](#)
  - ctrlr\_data\_msgs, [237](#)
  - ctrlr\_rx\_msgs, [237](#)
  - rx\_bytes\_sent, [238](#)
  - rxs\_sent, [238](#)
  - txw\_bytes, [238](#)
  - txw\_msgs, [238](#)
- lbm\_src\_transport\_stats\_lbtru\_t\_stct, [240](#)
  - bytes\_sent, [240](#)
  - msgs\_sent, [240](#)
  - nak\_pkts\_rcved, [240](#)
  - naks\_ignored, [240](#)
  - naks\_rcved, [241](#)
  - naks\_rx\_delay\_ignored, [241](#)
  - naks\_shed, [241](#)

- num\_clients, [241](#)
- rx\_bytes\_sent, [241](#)
- rxs\_sent, [242](#)
- lbm\_src\_transport\_stats\_lbtsmx\_t\_stct, [243](#)
  - bytes\_sent, [243](#)
  - msgs\_sent, [243](#)
  - num\_clients, [243](#)
- lbm\_src\_transport\_stats\_t
  - lbm.h, [424](#)
- lbm\_src\_transport\_stats\_t\_stct, [244](#)
  - daemon, [245](#)
  - lbtipc, [245](#)
  - lbtrdma, [245](#)
  - lbtrm, [245](#)
  - lbtru, [245](#)
  - lbtsmx, [245](#)
  - source, [245](#)
  - tcp, [245](#)
  - type, [245](#)
- lbm\_src\_transport\_stats\_tcp\_t\_stct, [247](#)
  - bytes\_buffered, [247](#)
  - num\_clients, [247](#)
- lbm\_src\_ume\_deregister
  - lbm.h, [540](#)
- lbm\_str\_hash\_func\_ex\_t
  - lbm.h, [424](#)
- lbm\_str\_hash\_func\_ex\_t\_stct, [248](#)
  - hashfunc, [248](#)
- lbm\_str\_hash\_function\_cb
  - lbm.h, [425](#)
- lbm\_str\_hash\_function\_cb\_ex
  - lbm.h, [425](#)
- lbm\_timer\_cb\_proc
  - lbm.h, [425](#)
- lbm\_timeval\_t
  - lbm.h, [426](#)
- lbm\_timeval\_t\_stct, [249](#)
- lbm\_topic\_from\_src
  - lbm.h, [540](#)
- LBM\_TOPIC\_RES\_REQUEST\_-ADVERTISEMENT
  - lbm.h, [405](#)
- LBM\_TOPIC\_RES\_REQUEST\_-CONTEXT\_-ADVERTISEMENT
  - lbm.h, [405](#)
- LBM\_TOPIC\_RES\_REQUEST\_-CONTEXT\_QUERY
  - lbm.h, [405](#)
- LBM\_TOPIC\_RES\_REQUEST\_GW\_-REMOTE\_INTEREST
  - lbm.h, [405](#)
- LBM\_TOPIC\_RES\_REQUEST\_QUERY
  - lbm.h, [405](#)
- LBM\_TOPIC\_RES\_REQUEST\_-RESERVED1
  - lbm.h, [405](#)
- LBM\_TOPIC\_RES\_REQUEST\_-WILDCARD\_QUERY
  - lbm.h, [405](#)
- lbm\_transport\_source\_format
  - lbm.h, [540](#)
- lbm\_transport\_source\_info\_t
  - lbm.h, [426](#)
- lbm\_transport\_source\_info\_t\_stct, [250](#)
  - dest\_port, [251](#)
  - mc\_group, [251](#)
  - session\_id, [251](#)
  - src\_ip, [251](#)
  - src\_port, [251](#)
  - topic\_idx, [251](#)
  - transport\_id, [251](#)
  - transport\_idx, [251](#)
  - type, [252](#)
- lbm\_transport\_source\_parse
  - lbm.h, [541](#)
- LBM\_TRANSPORT\_STAT\_DAEMON
  - lbm.h, [405](#)
- LBM\_TRANSPORT\_STAT\_LBTIPC
  - lbm.h, [406](#)
- LBM\_TRANSPORT\_STAT\_LBTRDMA
  - lbm.h, [406](#)
- LBM\_TRANSPORT\_STAT\_LBTRM
  - lbm.h, [406](#)
- LBM\_TRANSPORT\_STAT\_LBTRU
  - lbm.h, [406](#)
- LBM\_TRANSPORT\_STAT\_LBTSMX
  - lbm.h, [406](#)
- LBM\_TRANSPORT\_STAT\_TCP
  - lbm.h, [406](#)
- LBM\_TRANSPORT\_TYPE\_LBTIPC

- lbm.h, [406](#)
- LBM\_TRANSPORT\_TYPE\_-  
LBTRDMA
  - lbm.h, [406](#)
- LBM\_TRANSPORT\_TYPE\_LBTRM
  - lbm.h, [407](#)
- LBM\_TRANSPORT\_TYPE\_LBTRU
  - lbm.h, [407](#)
- LBM\_TRANSPORT\_TYPE\_LBTSMX
  - lbm.h, [407](#)
- LBM\_TRANSPORT\_TYPE\_TCP
  - lbm.h, [407](#)
- lbm\_ucast\_resolver\_entry\_t
  - lbm.h, [426](#)
- lbm\_ucast\_resolver\_entry\_t\_stct, [253](#)
  - destination\_port, [253](#)
  - iface, [253](#)
  - resolver\_ip, [253](#)
  - source\_port, [253](#)
- lbm\_ume\_ack\_delete
  - lbm.h, [541](#)
- lbm\_ume\_ack\_send\_explicit\_ack
  - lbm.h, [541](#)
- lbm\_ume\_ctx\_rcv\_ctx\_notification\_-  
create\_function\_cb
  - lbm.h, [427](#)
- lbm\_ume\_ctx\_rcv\_ctx\_notification\_-  
delete\_function\_cb
  - lbm.h, [427](#)
- lbm\_ume\_ctx\_rcv\_ctx\_notification\_-  
func\_t
  - lbm.h, [428](#)
- lbm\_ume\_ctx\_rcv\_ctx\_notification\_-  
func\_t\_stct, [255](#)
- lbm\_ume\_rcv\_recovery\_info\_ex\_func\_-  
info\_t
  - lbm.h, [428](#)
- lbm\_ume\_rcv\_recovery\_info\_ex\_func\_-  
info\_t\_stct, [256](#)
  - flags, [256](#)
  - high\_sequence\_number, [256](#)
  - low\_rxreq\_max\_sequence\_number,  
[256](#)
  - low\_sequence\_number, [257](#)
  - source, [257](#)
  - source\_clientid, [257](#)
  - src\_session\_id, [257](#)
- lbm\_ume\_rcv\_recovery\_info\_ex\_func\_t
  - lbm.h, [428](#)
- lbm\_ume\_rcv\_recovery\_info\_ex\_func\_-  
t\_stct, [258](#)
- lbm\_ume\_rcv\_recovery\_info\_ex\_-  
function\_cb
  - lbm.h, [428](#)
- lbm\_ume\_rcv\_regid\_ex\_func\_info\_t
  - lbm.h, [429](#)
- lbm\_ume\_rcv\_regid\_ex\_func\_info\_t\_-  
stct, [259](#)
  - flags, [259](#)
  - source, [259](#)
  - source\_clientid, [259](#)
  - src\_registration\_id, [259](#)
  - store, [260](#)
  - store\_index, [260](#)
- lbm\_ume\_rcv\_regid\_ex\_func\_t
  - lbm.h, [429](#)
- lbm\_ume\_rcv\_regid\_ex\_func\_t\_stct, [261](#)
- lbm\_ume\_rcv\_regid\_ex\_function\_cb
  - lbm.h, [429](#)
- lbm\_ume\_rcv\_regid\_func\_t
  - lbm.h, [429](#)
- lbm\_ume\_rcv\_regid\_func\_t\_stct, [262](#)
- lbm\_ume\_rcv\_regid\_function\_cb
  - lbm.h, [429](#)
- lbm\_ume\_src\_force\_reclaim\_func\_t
  - lbm.h, [430](#)
- lbm\_ume\_src\_force\_reclaim\_func\_t\_stct,  
[263](#)
- lbm\_ume\_src\_force\_reclaim\_function\_-  
cb
  - lbm.h, [430](#)
- lbm\_ume\_src\_msg\_stable
  - lbm.h, [542](#)
- lbm\_ume\_store\_entry\_t
  - lbm.h, [431](#)
- lbm\_ume\_store\_entry\_t\_stct, [264](#)
  - domain\_id, [264](#)
  - group\_index, [264](#)
  - ip\_address, [264](#)
  - registration\_id, [264](#)
  - tcp\_port, [264](#)
- lbm\_ume\_store\_group\_entry\_t

- lbm.h, 431
- lbm\_ume\_store\_group\_entry\_t\_stct, 266
  - group\_size, 266
  - index, 266
- lbm\_ume\_store\_name\_entry\_t
  - lbm.h, 431
- lbm\_ume\_store\_name\_entry\_t\_stct, 267
  - group\_index, 267
  - name, 267
  - registration\_id, 267
- LBM\_UMM\_INFO\_FLAGS\_USE\_SSL
  - lbm.h, 407
- lbm\_umm\_info\_t\_stct, 268
  - appname, 268
  - cert\_file, 268
  - cert\_file\_password, 268
  - flags, 268
  - password, 268
  - servers, 268
  - username, 269
- lbm\_umq\_ctx\_msg\_stable
  - lbm.h, 542
- LBM\_UMQ\_INDEX\_FLAG\_-
  - NUMERIC
  - lbm.h, 407
- lbm\_umq\_index\_info\_t
  - lbm.h, 431
- lbm\_umq\_index\_info\_t\_stct, 270
  - flags, 270
  - index, 270
  - index\_len, 270
- lbm\_umq\_msg\_selector\_create
  - lbm.h, 542
- lbm\_umq\_msg\_selector\_delete
  - lbm.h, 543
- lbm\_umq\_msg\_total\_lifetime\_info\_t
  - lbm.h, 431
- lbm\_umq\_msg\_total\_lifetime\_info\_t\_-
  - stct, 271
  - flags, 271
  - umq\_msg\_total\_lifetime, 271
- lbm\_umq\_msgid\_t
  - lbm.h, 431
- lbm\_umq\_msgid\_t\_stct, 272
  - regid, 272
  - stamp, 272
- lbm\_umq\_queue\_entry\_t
  - lbm.h, 431
- lbm\_umq\_queue\_entry\_t\_stct, 273
  - name, 273
  - regid, 273
- lbm\_umq\_queue\_msg\_status\_t, 274
  - clientd, 274
  - flags, 274
  - msg, 274
  - msgid, 274
  - status, 274
- lbm\_umq\_queue\_topic\_status\_t, 276
  - flags, 276
  - status, 276
  - topic, 276
- lbm\_umq\_queue\_topic\_t\_stct, 277
  - appsets, 277
  - num\_appsets, 277
  - reserved, 277
  - topic\_name, 277
- lbm\_umq\_regid\_t
  - lbm.h, 431
- lbm\_umq\_ulb\_application\_set\_attr\_t
  - lbm.h, 432
- lbm\_umq\_ulb\_application\_set\_attr\_t\_-
  - stct, 278
  - d, 278
  - index, 278
  - lu, 278
  - value, 278
- lbm\_umq\_ulb\_receiver\_type\_attr\_t
  - lbm.h, 432
- lbm\_umq\_ulb\_receiver\_type\_attr\_t\_stct,
  - 279
  - d, 279
  - id, 279
  - lu, 279
  - value, 279
- lbm\_umq\_ulb\_receiver\_type\_entry\_t
  - lbm.h, 432
- lbm\_umq\_ulb\_receiver\_type\_entry\_t\_-
  - stct, 280
  - application\_set\_index, 280
  - id, 280
- lbm\_unicast\_immediate\_message
  - lbm.h, 543

- lbm\_unicast\_immediate\_request  
lbm.h, 543
- lbm\_version  
lbm.h, 544
- lbm\_wildcard\_rcv\_attr\_create  
lbm.h, 544
- lbm\_wildcard\_rcv\_attr\_create\_default  
lbm.h, 545
- lbm\_wildcard\_rcv\_attr\_create\_from\_xml  
lbm.h, 545
- lbm\_wildcard\_rcv\_attr\_delete  
lbm.h, 546
- lbm\_wildcard\_rcv\_attr\_dump  
lbm.h, 546
- lbm\_wildcard\_rcv\_attr\_dup  
lbm.h, 546
- lbm\_wildcard\_rcv\_attr\_getopt  
lbm.h, 547
- lbm\_wildcard\_rcv\_attr\_option\_size  
lbm.h, 547
- lbm\_wildcard\_rcv\_attr\_set\_from\_xml  
lbm.h, 547
- lbm\_wildcard\_rcv\_attr\_setopt  
lbm.h, 548
- lbm\_wildcard\_rcv\_attr\_str\_getopt  
lbm.h, 548
- lbm\_wildcard\_rcv\_attr\_str\_setopt  
lbm.h, 549
- lbm\_wildcard\_rcv\_compare\_func\_t  
lbm.h, 432
- lbm\_wildcard\_rcv\_compare\_func\_t\_stct,  
281
- lbm\_wildcard\_rcv\_compare\_function\_cb  
lbm.h, 432
- lbm\_wildcard\_rcv\_create  
lbm.h, 549
- lbm\_wildcard\_rcv\_create\_func\_t  
lbm.h, 433
- lbm\_wildcard\_rcv\_create\_func\_t\_stct,  
282
- lbm\_wildcard\_rcv\_create\_function\_cb  
lbm.h, 433
- lbm\_wildcard\_rcv\_delete  
lbm.h, 550
- lbm\_wildcard\_rcv\_delete\_ex  
lbm.h, 550
- lbm\_wildcard\_rcv\_delete\_func\_t  
lbm.h, 433
- lbm\_wildcard\_rcv\_delete\_func\_t\_stct,  
283
- lbm\_wildcard\_rcv\_delete\_function\_cb  
lbm.h, 433
- lbm\_wildcard\_rcv\_dump  
lbm.h, 551
- lbm\_wildcard\_rcv\_getopt  
lbm.h, 551
- LBM\_WILDCARD\_RCV\_PATTERN\_-  
TYPE\_APP\_CB  
lbm.h, 407
- LBM\_WILDCARD\_RCV\_PATTERN\_-  
TYPE\_PCRE  
lbm.h, 407
- LBM\_WILDCARD\_RCV\_PATTERN\_-  
TYPE\_REGEX  
lbm.h, 407
- lbm\_wildcard\_rcv\_setopt  
lbm.h, 551
- lbm\_wildcard\_rcv\_stats\_t  
lbm.h, 434
- lbm\_wildcard\_rcv\_stats\_t\_stct, 284  
pattern, 284  
type, 284
- lbm\_wildcard\_rcv\_str\_getopt  
lbm.h, 552
- lbm\_wildcard\_rcv\_str\_setopt  
lbm.h, 552
- lbm\_wildcard\_rcv\_subscribe\_channel  
lbm.h, 553
- lbm\_wildcard\_rcv\_umq\_deregister  
lbm.h, 553
- lbm\_wildcard\_rcv\_umq\_index\_release  
lbm.h, 553
- lbm\_wildcard\_rcv\_umq\_index\_start\_-  
assignment  
lbm.h, 554
- lbm\_wildcard\_rcv\_umq\_index\_stop\_-  
assignment  
lbm.h, 554
- lbm\_wildcard\_rcv\_unsubscribe\_channel  
lbm.h, 554
- lbm\_wildcard\_rcv\_unsubscribe\_-  
channel\_ex

- lbm.h, [555](#)
- lbm\_win32\_static\_thread\_attach
  - lbm.h, [555](#)
- lbm\_win32\_static\_thread\_detach
  - lbm.h, [555](#)
- lbm\_wrcv\_ume\_deregister
  - lbm.h, [556](#)
- lbmaux.h, [557](#)
  - lbmaux\_context\_attr\_setopt\_from\_file, [558](#)
  - lbmaux\_context\_create\_from\_file, [559](#)
  - lbmaux\_event\_queue\_attr\_setopt\_from\_file, [559](#)
  - lbmaux\_rcv\_topic\_attr\_setopt\_from\_file, [559](#)
  - lbmaux\_src\_topic\_attr\_setopt\_from\_file, [560](#)
  - lbmaux\_wildcard\_rcv\_attr\_setopt\_from\_file, [560](#)
- lbmaux\_context\_attr\_setopt\_from\_file
  - lbmaux.h, [558](#)
- lbmaux\_context\_create\_from\_file
  - lbmaux.h, [559](#)
- lbmaux\_event\_queue\_attr\_setopt\_from\_file
  - lbmaux.h, [559](#)
- lbmaux\_rcv\_topic\_attr\_setopt\_from\_file
  - lbmaux.h, [559](#)
- lbmaux\_src\_topic\_attr\_setopt\_from\_file
  - lbmaux.h, [560](#)
- lbmaux\_wildcard\_rcv\_attr\_setopt\_from\_file
  - lbmaux.h, [560](#)
- lbmht.h, [561](#)
  - lbm\_delete\_cb\_proc, [564](#)
  - lbm\_hypertopic\_rcv\_add, [565](#)
  - lbm\_hypertopic\_rcv\_cb\_proc, [564](#)
  - lbm\_hypertopic\_rcv\_delete, [565](#)
  - lbm\_hypertopic\_rcv\_destroy, [565](#)
  - lbm\_hypertopic\_rcv\_init, [566](#)
- lbmmon.h, [567](#)
  - LBMMON\_ATTR\_APPSOURCEID, [576](#)
  - LBMMON\_ATTR\_CONTEXTID, [576](#)
  - LBMMON\_ATTR\_CTXINST, [577](#)
  - LBMMON\_ATTR\_DOMAINID, [577](#)
  - LBMMON\_ATTR\_FORMAT\_MODULEID, [577](#)
  - lbmmon\_attr\_get\_appsourceid, [594](#)
  - lbmmon\_attr\_get\_contextid, [594](#)
  - lbmmon\_attr\_get\_ctxinst, [594](#)
  - lbmmon\_attr\_get\_domainid, [595](#)
  - lbmmon\_attr\_get\_ipv4sender, [595](#)
  - lbmmon\_attr\_get\_objectid, [595](#)
  - lbmmon\_attr\_get\_processid, [595](#)
  - lbmmon\_attr\_get\_source, [596](#)
  - lbmmon\_attr\_get\_timestamp, [596](#)
  - LBMMON\_ATTR\_OBJECTID, [577](#)
  - LBMMON\_ATTR\_PROCESSID, [577](#)
  - LBMMON\_ATTR\_SENDER\_IPV4, [577](#)
  - LBMMON\_ATTR\_SOURCE, [577](#)
  - LBMMON\_ATTR\_SOURCE\_IM, [577](#)
  - LBMMON\_ATTR\_SOURCE\_NORMAL, [577](#)
  - LBMMON\_ATTR\_TIMESTAMP, [578](#)
  - lbmmon\_context\_monitor, [596](#)
  - lbmmon\_context\_unmonitor, [597](#)
  - lbmmon\_ctx\_format\_deserialize\_t, [580](#)
  - lbmmon\_ctx\_format\_serialize\_t, [580](#)
  - lbmmon\_ctx\_statistics\_cb, [581](#)
  - lbmmon\_ctx\_statistics\_func\_t, [581](#)
  - LBMMON\_EAGAIN, [578](#)
  - LBMMON\_EALREADY, [578](#)
  - LBMMON\_EINVAL, [578](#)
  - LBMMON\_ELBMFAIL, [578](#)
  - LBMMON\_EMODFAIL, [578](#)
  - LBMMON\_ENOMEM, [578](#)
  - LBMMON\_EOP, [578](#)
  - lbmmon\_errmsg, [597](#)
  - lbmmon\_errnum, [597](#)
  - LBMMON\_ERROR\_BASE, [578](#)
  - lbmmon\_evq\_format\_deserialize\_t, [581](#)

- lbmmon\_evq\_format\_serialize\_t, 582
- lbmmon\_evq\_monitor, 597
- lbmmon\_evq\_statistics\_cb, 583
- lbmmon\_evq\_statistics\_func\_t, 583
- lbmmon\_evq\_unmonitor, 598
- lbmmon\_format\_errmsg\_t, 583
- lbmmon\_format\_finish\_t, 583
- lbmmon\_format\_init\_t, 584
- lbmmon\_next\_key\_value\_pair, 598
- lbmmon\_packet\_hdr\_t, 584
- LBMMON\_PACKET\_-  
SIGNATURE, 578
- LBMMON\_PACKET\_TYPE\_-  
CONTEXT, 579
- LBMMON\_PACKET\_TYPE\_-  
EVENT\_QUEUE, 579
- LBMMON\_PACKET\_TYPE\_-  
RECEIVER, 579
- LBMMON\_PACKET\_TYPE\_-  
RECEIVER\_TOPIC, 579
- LBMMON\_PACKET\_TYPE\_-  
SOURCE, 579
- LBMMON\_PACKET\_TYPE\_-  
WILDCARD\_RECEIVER, 579
- lbmmon\_rctl\_attr\_create, 599
- lbmmon\_rctl\_attr\_delete, 599
- lbmmon\_rctl\_attr\_getopt, 599
- lbmmon\_rctl\_attr\_setopt, 600
- LBMMON\_RCTL\_CONTEXT\_-  
CALLBACK, 579
- lbmmon\_rctl\_create, 600
- lbmmon\_rctl\_destroy, 601
- LBMMON\_RCTL\_EVENT\_-  
QUEUE\_CALLBACK, 579
- LBMMON\_RCTL\_RECEIVER\_-  
CALLBACK, 579
- LBMMON\_RCTL\_RECEIVER\_-  
TOPIC\_CALLBACK, 579
- LBMMON\_RCTL\_SOURCE\_-  
CALLBACK, 580
- LBMMON\_RCTL\_WILDCARD\_-  
RECEIVER\_CALLBACK, 580
- lbmmon\_rcv\_format\_deserialize\_t, 584
- lbmmon\_rcv\_format\_serialize\_t, 585
- lbmmon\_rcv\_monitor, 601
- lbmmon\_rcv\_statistics\_cb, 586
- lbmmon\_rcv\_statistics\_func\_t, 586
- lbmmon\_rcv\_topic\_format\_-  
deserialize\_t, 586
- lbmmon\_rcv\_topic\_format\_-  
serialize\_t, 587
- lbmmon\_rcv\_topic\_statistics\_cb, 588
- lbmmon\_rcv\_topic\_statistics\_func\_-  
t, 588
- lbmmon\_rcv\_unmonitor, 602
- lbmmon\_sctl\_create, 602
- lbmmon\_sctl\_destroy, 603
- lbmmon\_sctl\_sample, 603
- lbmmon\_sctl\_sample\_ex, 603
- lbmmon\_src\_format\_deserialize\_t, 588
- lbmmon\_src\_format\_serialize\_t, 589
- lbmmon\_src\_monitor, 603
- lbmmon\_src\_statistics\_cb, 589
- lbmmon\_src\_statistics\_func\_t, 590
- lbmmon\_src\_unmonitor, 604
- lbmmon\_transport\_errmsg\_t, 590
- lbmmon\_transport\_finishrcv\_t, 590
- lbmmon\_transport\_finishsrc\_t, 590
- lbmmon\_transport\_initrcv\_t, 591
- lbmmon\_transport\_initsrc\_t, 591
- lbmmon\_transport\_receive\_t, 591
- lbmmon\_transport\_send\_t, 592
- lbmmon\_wildcard\_rcv\_format\_-  
deserialize\_t, 592
- lbmmon\_wildcard\_rcv\_format\_-  
serialize\_t, 592
- lbmmon\_wildcard\_rcv\_statistics\_cb, 593
- lbmmon\_wildcard\_rcv\_statistics\_-  
func\_t, 593
- LBMMON\_ATTR\_APPSOURCEID  
lbmmon.h, 576
- lbmmon\_attr\_block\_t\_stct, 285
- mEntryCount, 285

- mEntryLength, [285](#)
- LBMMON\_ATTR\_CONTEXTID
  - lbmmon.h, [576](#)
- LBMMON\_ATTR\_CTXINST
  - lbmmon.h, [577](#)
- LBMMON\_ATTR\_DOMAINID
  - lbmmon.h, [577](#)
- lbmmon\_attr\_entry\_t\_stct, [286](#)
  - mLength, [286](#)
  - mType, [286](#)
- LBMMON\_ATTR\_FORMAT\_-MODULEID
  - lbmmon.h, [577](#)
- lbmmon\_attr\_get\_appsourceid
  - lbmmon.h, [594](#)
- lbmmon\_attr\_get\_contextid
  - lbmmon.h, [594](#)
- lbmmon\_attr\_get\_ctxinst
  - lbmmon.h, [594](#)
- lbmmon\_attr\_get\_domainid
  - lbmmon.h, [595](#)
- lbmmon\_attr\_get\_ipv4sender
  - lbmmon.h, [595](#)
- lbmmon\_attr\_get\_objectid
  - lbmmon.h, [595](#)
- lbmmon\_attr\_get\_processid
  - lbmmon.h, [595](#)
- lbmmon\_attr\_get\_source
  - lbmmon.h, [596](#)
- lbmmon\_attr\_get\_timestamp
  - lbmmon.h, [596](#)
- LBMMON\_ATTR\_OBJECTID
  - lbmmon.h, [577](#)
- LBMMON\_ATTR\_PROCESSID
  - lbmmon.h, [577](#)
- LBMMON\_ATTR\_SENDER\_IPV4
  - lbmmon.h, [577](#)
- LBMMON\_ATTR\_SOURCE
  - lbmmon.h, [577](#)
- LBMMON\_ATTR\_SOURCE\_IM
  - lbmmon.h, [577](#)
- LBMMON\_ATTR\_SOURCE\_-NORMAL
  - lbmmon.h, [577](#)
- LBMMON\_ATTR\_TIMESTAMP
  - lbmmon.h, [578](#)
- lbmmon\_context\_monitor
  - lbmmon.h, [596](#)
- lbmmon\_context\_unmonitor
  - lbmmon.h, [597](#)
- lbmmon\_ctx\_format\_deserialize\_t
  - lbmmon.h, [580](#)
- lbmmon\_ctx\_format\_serialize\_t
  - lbmmon.h, [580](#)
- lbmmon\_ctx\_statistics\_cb
  - lbmmon.h, [581](#)
- lbmmon\_ctx\_statistics\_func\_t
  - lbmmon.h, [581](#)
- lbmmon\_ctx\_statistics\_func\_t\_stct, [287](#)
  - cbfunc, [287](#)
- LBMMON\_EAGAIN
  - lbmmon.h, [578](#)
- LBMMON\_EALREADY
  - lbmmon.h, [578](#)
- LBMMON\_EINVAL
  - lbmmon.h, [578](#)
- LBMMON\_ELBMFAIL
  - lbmmon.h, [578](#)
- LBMMON\_EMODFAIL
  - lbmmon.h, [578](#)
- LBMMON\_ENOMEM
  - lbmmon.h, [578](#)
- LBMMON\_EOP
  - lbmmon.h, [578](#)
- lbmmon\_errmsg
  - lbmmon.h, [597](#)
- lbmmon\_errnum
  - lbmmon.h, [597](#)
- LBMMON\_ERROR\_BASE
  - lbmmon.h, [578](#)
- lbmmon\_evq\_format\_deserialize\_t
  - lbmmon.h, [581](#)
- lbmmon\_evq\_format\_serialize\_t
  - lbmmon.h, [582](#)
- lbmmon\_evq\_monitor
  - lbmmon.h, [597](#)
- lbmmon\_evq\_statistics\_cb
  - lbmmon.h, [583](#)
- lbmmon\_evq\_statistics\_func\_t
  - lbmmon.h, [583](#)
- lbmmon\_evq\_statistics\_func\_t\_stct, [288](#)
  - cbfunc, [288](#)

- lbmmon\_evq\_unmonitor
  - lbmmon.h, [598](#)
- lbmmon\_format\_errmsg\_t
  - lbmmon.h, [583](#)
- lbmmon\_format\_finish\_t
  - lbmmon.h, [583](#)
- lbmmon\_format\_func\_t\_stct, [289](#)
  - mCtxDeserialize, [289](#)
  - mCtxSerialize, [289](#)
  - mErrorMessage, [289](#)
  - mEvqDeserialize, [289](#)
  - mEvqSerialize, [290](#)
  - mFinish, [290](#)
  - mInit, [290](#)
  - mRcvDeserialize, [290](#)
  - mRcvSerialize, [290](#)
  - mRcvTopicDeserialize, [290](#)
  - mRcvTopicSerialize, [290](#)
  - mSrcDeserialize, [290](#)
  - mSrcSerialize, [291](#)
  - mWildcardRcvDeserialize, [291](#)
  - mWildcardRcvSerialize, [291](#)
- lbmmon\_format\_init\_t
  - lbmmon.h, [584](#)
- lbmmon\_next\_key\_value\_pair
  - lbmmon.h, [598](#)
- lbmmon\_packet\_hdr\_t
  - lbmmon.h, [584](#)
- lbmmon\_packet\_hdr\_t\_stct, [292](#)
  - mAttributeBlockLength, [292](#)
  - mDataLength, [292](#)
  - mFiller, [292](#)
  - mSignature, [292](#)
  - mType, [292](#)
- LBMMON\_PACKET\_SIGNATURE
  - lbmmon.h, [578](#)
- LBMMON\_PACKET\_TYPE\_-  
CONTEXT
  - lbmmon.h, [579](#)
- LBMMON\_PACKET\_TYPE\_EVENT\_-  
QUEUE
  - lbmmon.h, [579](#)
- LBMMON\_PACKET\_TYPE\_-  
RECEIVER
  - lbmmon.h, [579](#)
- LBMMON\_PACKET\_TYPE\_-  
RECEIVER\_TOPIC
  - lbmmon.h, [579](#)
- LBMMON\_PACKET\_TYPE\_SOURCE
  - lbmmon.h, [579](#)
- LBMMON\_PACKET\_TYPE\_-  
WILDCARD\_RECEIVER
  - lbmmon.h, [579](#)
- lbmmon\_rctl\_attr\_create
  - lbmmon.h, [599](#)
- lbmmon\_rctl\_attr\_delete
  - lbmmon.h, [599](#)
- lbmmon\_rctl\_attr\_getopt
  - lbmmon.h, [599](#)
- lbmmon\_rctl\_attr\_setopt
  - lbmmon.h, [600](#)
- LBMMON\_RCTL\_CONTEXT\_-  
CALLBACK
  - lbmmon.h, [579](#)
- lbmmon\_rctl\_create
  - lbmmon.h, [600](#)
- lbmmon\_rctl\_destroy
  - lbmmon.h, [601](#)
- LBMMON\_RCTL\_EVENT\_QUEUE\_-  
CALLBACK
  - lbmmon.h, [579](#)
- LBMMON\_RCTL\_RECEIVER\_-  
CALLBACK
  - lbmmon.h, [579](#)
- LBMMON\_RCTL\_RECEIVER\_-  
TOPIC\_CALLBACK
  - lbmmon.h, [579](#)
- LBMMON\_RCTL\_SOURCE\_-  
CALLBACK
  - lbmmon.h, [580](#)
- LBMMON\_RCTL\_WILDCARD\_-  
RECEIVER\_CALLBACK
  - lbmmon.h, [580](#)
- lbmmon\_rcv\_format\_deserialize\_t
  - lbmmon.h, [584](#)
- lbmmon\_rcv\_format\_serialize\_t
  - lbmmon.h, [585](#)
- lbmmon\_rcv\_monitor
  - lbmmon.h, [601](#)
- lbmmon\_rcv\_statistics\_cb
  - lbmmon.h, [586](#)

- lbmmon\_rcv\_statistics\_func\_t
  - lbmmon.h, 586
- lbmmon\_rcv\_statistics\_func\_t\_stct, 294
  - cbfunc, 294
- lbmmon\_rcv\_topic\_format\_deserialize\_t
  - lbmmon.h, 586
- lbmmon\_rcv\_topic\_format\_serialize\_t
  - lbmmon.h, 587
- lbmmon\_rcv\_topic\_statistics\_cb
  - lbmmon.h, 588
- lbmmon\_rcv\_topic\_statistics\_func\_t
  - lbmmon.h, 588
- lbmmon\_rcv\_topic\_statistics\_func\_t\_stct, 295
  - cbfunc, 295
- lbmmon\_rcv\_unmonitor
  - lbmmon.h, 602
- lbmmon\_sctl\_create
  - lbmmon.h, 602
- lbmmon\_sctl\_destroy
  - lbmmon.h, 603
- lbmmon\_sctl\_sample
  - lbmmon.h, 603
- lbmmon\_sctl\_sample\_ex
  - lbmmon.h, 603
- lbmmon\_src\_format\_deserialize\_t
  - lbmmon.h, 588
- lbmmon\_src\_format\_serialize\_t
  - lbmmon.h, 589
- lbmmon\_src\_monitor
  - lbmmon.h, 603
- lbmmon\_src\_statistics\_cb
  - lbmmon.h, 589
- lbmmon\_src\_statistics\_func\_t
  - lbmmon.h, 590
- lbmmon\_src\_statistics\_func\_t\_stct, 296
  - cbfunc, 296
- lbmmon\_src\_unmonitor
  - lbmmon.h, 604
- lbmmon\_transport\_errmsg\_t
  - lbmmon.h, 590
- lbmmon\_transport\_finishrcv\_t
  - lbmmon.h, 590
- lbmmon\_transport\_finishsrc\_t
  - lbmmon.h, 590
- lbmmon\_transport\_func\_t\_stct, 297
  - mErrorMessage, 297
  - mFinishReceiver, 297
  - mFinishSource, 297
  - mInitReceiver, 297
  - mInitSource, 297
  - mReceive, 297
  - mSend, 298
- lbmmon\_transport\_initrcv\_t
  - lbmmon.h, 591
- lbmmon\_transport\_initsrc\_t
  - lbmmon.h, 591
- lbmmon\_transport\_receive\_t
  - lbmmon.h, 591
- lbmmon\_transport\_send\_t
  - lbmmon.h, 592
- lbmmon\_wildcard\_rcv\_format\_-
  - deserialize\_t
  - lbmmon.h, 592
- lbmmon\_wildcard\_rcv\_format\_-
  - serialize\_t
  - lbmmon.h, 592
- lbmmon\_wildcard\_rcv\_statistics\_cb
  - lbmmon.h, 593
- lbmmon\_wildcard\_rcv\_statistics\_func\_t
  - lbmmon.h, 593
- lbmmon\_wildcard\_rcv\_statistics\_func\_-
  - t\_stct, 299
  - cbfunc, 299
- lbmpdm.h, 605
  - lbmpdm\_cache\_init, 627
  - lbmpdm\_cache\_struct\_add, 627
  - lbmpdm\_cache\_struct\_find, 628
  - lbmpdm\_cache\_struct\_find\_by\_-
    - version, 628
  - lbmpdm\_cache\_struct\_remove, 628
  - lbmpdm\_cache\_struct\_remove\_by\_-
    - version, 628
  - lbmpdm\_defn\_add\_field\_info\_by\_-
    - int\_name, 628
  - lbmpdm\_defn\_add\_field\_info\_by\_-
    - str\_name, 629
  - lbmpdm\_defn\_create, 629
  - lbmpdm\_defn\_delete, 630
  - lbmpdm\_defn\_deserialize, 630
  - lbmpdm\_defn\_finalize, 631

- lbmpdm\_defn\_get\_field\_handle\_-  
by\_int\_name, [631](#)
- lbmpdm\_defn\_get\_field\_handle\_-  
by\_str\_name, [631](#)
- lbmpdm\_defn\_get\_field\_info\_int\_-  
name, [631](#)
- lbmpdm\_defn\_get\_field\_info\_str\_-  
name, [632](#)
- lbmpdm\_defn\_get\_field\_info\_type,  
[632](#)
- lbmpdm\_defn\_get\_field\_names\_-  
type, [632](#)
- lbmpdm\_defn\_get\_id, [632](#)
- lbmpdm\_defn\_get\_length, [633](#)
- lbmpdm\_defn\_get\_msg\_vers\_major,  
[633](#)
- lbmpdm\_defn\_get\_msg\_vers\_-  
minor, [633](#)
- lbmpdm\_defn\_get\_num\_fields, [633](#)
- lbmpdm\_defn\_is\_finalized, [634](#)
- lbmpdm\_defn\_serialize, [634](#)
- lbmpdm\_errmsg, [634](#)
- lbmpdm\_errnum, [634](#)
- lbmpdm\_field\_value\_stct\_delete,  
[634](#)
- lbmpdm\_iter\_create, [635](#)
- lbmpdm\_iter\_create\_from\_field\_-  
handle, [635](#)
- lbmpdm\_iter\_delete, [635](#)
- lbmpdm\_iter\_first, [636](#)
- lbmpdm\_iter\_get\_current, [636](#)
- lbmpdm\_iter\_get\_current\_field\_-  
value, [636](#)
- lbmpdm\_iter\_get\_current\_field\_-  
value\_vec, [636](#)
- lbmpdm\_iter\_has\_next, [637](#)
- lbmpdm\_iter\_is\_current\_set, [637](#)
- lbmpdm\_iter\_next, [637](#)
- lbmpdm\_iter\_set\_current\_field\_-  
value, [638](#)
- lbmpdm\_iter\_set\_current\_field\_-  
value\_vec, [638](#)
- lbmpdm\_iter\_set\_msg, [638](#)
- lbmpdm\_msg\_and\_defn\_delete, [638](#)
- lbmpdm\_msg\_create, [639](#)
- lbmpdm\_msg\_delete, [639](#)
- lbmpdm\_msg\_deserialize, [639](#)
- lbmpdm\_msg\_get\_data, [640](#)
- lbmpdm\_msg\_get\_defn, [640](#)
- lbmpdm\_msg\_get\_field\_value, [640](#)
- lbmpdm\_msg\_get\_field\_value\_stct,  
[641](#)
- lbmpdm\_msg\_get\_field\_value\_vec,  
[641](#)
- lbmpdm\_msg\_get\_length, [642](#)
- lbmpdm\_msg\_is\_field\_set, [642](#)
- lbmpdm\_msg\_remove\_field\_value,  
[642](#)
- lbmpdm\_msg\_serialize, [642](#)
- lbmpdm\_msg\_set\_field\_value, [643](#)
- lbmpdm\_msg\_set\_field\_value\_vec,  
[643](#)
- lbmpdm\_msg\_set\_incl\_defn\_flag,  
[643](#)
- lbmpdm\_msg\_unset\_incl\_defn\_flag,  
[644](#)
- PDM\_DEFN\_INT\_FIELD\_-  
NAMES, [620](#)
- PDM\_DEFN\_STR\_FIELD\_-  
NAMES, [620](#)
- PDM\_ERR\_CREATE\_BUFFER,  
[620](#)
- PDM\_ERR\_CREATE\_SECTION,  
[621](#)
- PDM\_ERR\_DEFN\_INVALID, [621](#)
- PDM\_ERR\_EINVAL, [621](#)
- PDM\_ERR\_FIELD\_IS\_NULL, [621](#)
- PDM\_ERR\_FIELD\_NOT\_FOUND,  
[621](#)
- PDM\_ERR\_INSUFFICIENT\_-  
BUFFER\_LENGTH, [621](#)
- PDM\_ERR\_MSG\_INVALID, [621](#)
- PDM\_ERR\_NO\_MORE\_FIELDS,  
[621](#)
- PDM\_ERR\_NOMEM, [621](#)
- PDM\_ERR\_REQ\_FIELD\_NOT\_-  
SET, [621](#)
- PDM\_FAILURE, [622](#)
- PDM\_FALSE, [622](#)
- PDM\_FIELD\_INFO\_FLAG\_-  
FIXED\_STR\_LEN, [622](#)

- PDM\_FIELD\_INFO\_FLAG\_-  
  NUM\_ARR\_ELEM, 622
- PDM\_FIELD\_INFO\_FLAG\_REQ,  
  622
- PDM\_INTERNAL\_TYPE\_-  
  INVALID, 622
- PDM\_ITER\_INVALID\_FIELD\_-  
  HANDLE, 622
- PDM\_MSG\_FLAG\_DEL\_DEFN\_-  
  WHEN\_REPLACED, 622
- PDM\_MSG\_FLAG\_INCL\_DEFN,  
  622
- PDM\_MSG\_FLAG\_NEED\_-  
  BYTE\_SWAP, 623
- PDM\_MSG\_FLAG\_TRY\_LOAD\_-  
  DEFN\_FROM\_CACHE, 623
- PDM\_MSG\_FLAG\_USE\_MSG\_-  
  DEFN\_IF\_NEEDED, 623
- PDM\_MSG\_FLAG\_VAR\_OR\_-  
  OPT\_FLDS\_SET, 623
- PDM\_MSG\_VER\_POLICY\_BEST,  
  623
- PDM\_MSG\_VER\_POLICY\_-  
  EXACT, 623
- PDM\_SUCCESS, 623
- PDM\_TRUE, 623
- PDM\_TYPE\_BLOB, 623
- PDM\_TYPE\_BLOB\_ARR, 624
- PDM\_TYPE\_BOOLEAN, 624
- PDM\_TYPE\_BOOLEAN\_ARR,  
  624
- PDM\_TYPE\_DECIMAL, 624
- PDM\_TYPE\_DECIMAL\_ARR, 624
- PDM\_TYPE\_DOUBLE, 624
- PDM\_TYPE\_DOUBLE\_ARR, 624
- PDM\_TYPE\_FIX\_STRING, 624
- PDM\_TYPE\_FIX\_STRING\_ARR,  
  624
- PDM\_TYPE\_FIX\_UNICODE, 624
- PDM\_TYPE\_FIX\_UNICODE\_-  
  ARR, 625
- PDM\_TYPE\_FLOAT, 625
- PDM\_TYPE\_FLOAT\_ARR, 625
- PDM\_TYPE\_INT16, 625
- PDM\_TYPE\_INT16\_ARR, 625
- PDM\_TYPE\_INT32, 625
- PDM\_TYPE\_INT32\_ARR, 625
- PDM\_TYPE\_INT64, 625
- PDM\_TYPE\_INT64\_ARR, 625
- PDM\_TYPE\_INT8, 625
- PDM\_TYPE\_INT8\_ARR, 626
- PDM\_TYPE\_MESSAGE, 626
- PDM\_TYPE\_MESSAGE\_ARR,  
  626
- PDM\_TYPE\_STRING, 626
- PDM\_TYPE\_STRING\_ARR, 626
- PDM\_TYPE\_TIMESTAMP, 626
- PDM\_TYPE\_TIMESTAMP\_ARR,  
  626
- PDM\_TYPE\_UINT16, 626
- PDM\_TYPE\_UINT16\_ARR, 626
- PDM\_TYPE\_UINT32, 626
- PDM\_TYPE\_UINT32\_ARR, 627
- PDM\_TYPE\_UINT64, 627
- PDM\_TYPE\_UINT64\_ARR, 627
- PDM\_TYPE\_UINT8, 627
- PDM\_TYPE\_UINT8\_ARR, 627
- PDM\_TYPE\_UNICODE, 627
- PDM\_TYPE\_UNICODE\_ARR, 627
- lbmpdm\_cache\_init  
  lbmpdm.h, 627
- lbmpdm\_cache\_struct\_add  
  lbmpdm.h, 627
- lbmpdm\_cache\_struct\_find  
  lbmpdm.h, 628
- lbmpdm\_cache\_struct\_find\_by\_version  
  lbmpdm.h, 628
- lbmpdm\_cache\_struct\_remove  
  lbmpdm.h, 628
- lbmpdm\_cache\_struct\_remove\_by\_-  
  version  
  lbmpdm.h, 628
- lbmpdm\_decimal\_t, 300  
  exp, 300  
  mant, 300
- lbmpdm\_defn\_add\_field\_info\_by\_int\_-  
  name  
  lbmpdm.h, 628
- lbmpdm\_defn\_add\_field\_info\_by\_str\_-  
  name  
  lbmpdm.h, 629
- lbmpdm\_defn\_create

- lbmpdm.h, 629
- lbmpdm\_defn\_delete
  - lbmpdm.h, 630
- lbmpdm\_defn\_deserialize
  - lbmpdm.h, 630
- lbmpdm\_defn\_finalize
  - lbmpdm.h, 631
- lbmpdm\_defn\_get\_field\_handle\_by\_int\_  
-name
  - lbmpdm.h, 631
- lbmpdm\_defn\_get\_field\_handle\_by\_str\_  
-name
  - lbmpdm.h, 631
- lbmpdm\_defn\_get\_field\_info\_int\_name
  - lbmpdm.h, 631
- lbmpdm\_defn\_get\_field\_info\_str\_name
  - lbmpdm.h, 632
- lbmpdm\_defn\_get\_field\_info\_type
  - lbmpdm.h, 632
- lbmpdm\_defn\_get\_field\_names\_type
  - lbmpdm.h, 632
- lbmpdm\_defn\_get\_id
  - lbmpdm.h, 632
- lbmpdm\_defn\_get\_length
  - lbmpdm.h, 633
- lbmpdm\_defn\_get\_msg\_vers\_major
  - lbmpdm.h, 633
- lbmpdm\_defn\_get\_msg\_vers\_minor
  - lbmpdm.h, 633
- lbmpdm\_defn\_get\_num\_fields
  - lbmpdm.h, 633
- lbmpdm\_defn\_is\_finalized
  - lbmpdm.h, 634
- lbmpdm\_defn\_serialize
  - lbmpdm.h, 634
- lbmpdm\_errmsg
  - lbmpdm.h, 634
- lbmpdm\_errnum
  - lbmpdm.h, 634
- lbmpdm\_field\_info\_attr\_stct\_t, 301
  - fixed\_str\_len, 301
  - num\_arr\_elem, 301
  - req, 301
- lbmpdm\_field\_value\_stct\_delete
  - lbmpdm.h, 634
- lbmpdm\_field\_value\_stct\_t, 302
  - field\_type, 302
  - is\_array, 302
  - is\_fixed, 302
  - len, 302
  - len\_arr, 302
  - num\_arr\_elem, 302
  - value, 303
  - value\_arr, 303
- lbmpdm\_iter\_create
  - lbmpdm.h, 635
- lbmpdm\_iter\_create\_from\_field\_handle
  - lbmpdm.h, 635
- lbmpdm\_iter\_delete
  - lbmpdm.h, 635
- lbmpdm\_iter\_first
  - lbmpdm.h, 636
- lbmpdm\_iter\_get\_current
  - lbmpdm.h, 636
- lbmpdm\_iter\_get\_current\_field\_value
  - lbmpdm.h, 636
- lbmpdm\_iter\_get\_current\_field\_value\_  
-vec
  - lbmpdm.h, 636
- lbmpdm\_iter\_has\_next
  - lbmpdm.h, 637
- lbmpdm\_iter\_is\_current\_set
  - lbmpdm.h, 637
- lbmpdm\_iter\_next
  - lbmpdm.h, 637
- lbmpdm\_iter\_set\_get\_current\_field\_value
  - lbmpdm.h, 638
- lbmpdm\_iter\_set\_get\_current\_field\_value\_  
-vec
  - lbmpdm.h, 638
- lbmpdm\_iter\_set\_msg
  - lbmpdm.h, 638
- lbmpdm\_msg\_and\_defn\_delete
  - lbmpdm.h, 638
- lbmpdm\_msg\_create
  - lbmpdm.h, 639
- lbmpdm\_msg\_delete
  - lbmpdm.h, 639
- lbmpdm\_msg\_deserialize
  - lbmpdm.h, 639
- lbmpdm\_msg\_get\_data
  - lbmpdm.h, 640

- lbmpdm\_msg\_get\_defn  
lbmpdm.h, 640
- lbmpdm\_msg\_get\_field\_value  
lbmpdm.h, 640
- lbmpdm\_msg\_get\_field\_value\_stct  
lbmpdm.h, 641
- lbmpdm\_msg\_get\_field\_value\_vec  
lbmpdm.h, 641
- lbmpdm\_msg\_get\_length  
lbmpdm.h, 642
- lbmpdm\_msg\_is\_field\_set  
lbmpdm.h, 642
- lbmpdm\_msg\_remove\_field\_value  
lbmpdm.h, 642
- lbmpdm\_msg\_serialize  
lbmpdm.h, 642
- lbmpdm\_msg\_set\_field\_value  
lbmpdm.h, 643
- lbmpdm\_msg\_set\_field\_value\_vec  
lbmpdm.h, 643
- lbmpdm\_msg\_set\_incl\_defn\_flag  
lbmpdm.h, 643
- lbmpdm\_msg\_unset\_incl\_defn\_flag  
lbmpdm.h, 644
- lbmpdm\_timestamp\_t, 304
  - tv\_secs, 304
  - tv\_usecs, 304
- lbmsdm.h, 645
  - lbmsdm\_decimal\_t, 681
  - LBMSDM\_ERR\_ADDING\_-  
FIELD, 684
  - LBMSDM\_ERR\_BAD\_TYPE, 683
  - LBMSDM\_ERR\_CANNOT\_-  
CONVERT, 683
  - LBMSDM\_ERR\_DELETING\_-  
FIELD, 684
  - LBMSDM\_ERR\_DUPLICATE\_-  
FIELD, 683
  - LBMSDM\_ERR\_EINVAL, 683
  - LBMSDM\_ERR\_ELEMENT\_-  
NOT\_FOUND, 684
  - LBMSDM\_ERR\_ENOMEM, 683
  - LBMSDM\_ERR\_FIELD\_IS\_-  
NULL, 684
  - LBMSDM\_ERR\_FIELD\_NOT\_-  
FOUND, 683
  - LBMSDM\_ERR\_INVALID\_-  
FIELD\_NAME, 684
  - LBMSDM\_ERR\_ITERATOR\_-  
INVALID, 684
  - LBMSDM\_ERR\_MSG\_INVALID,  
683
  - LBMSDM\_ERR\_-  
NAMETOOLONG, 683
  - LBMSDM\_ERR\_NOT\_ARRAY,  
683
  - LBMSDM\_ERR\_NOT\_SCALAR,  
684
  - LBMSDM\_ERR\_TYPE\_-  
MISMATCH, 684
  - LBMSDM\_ERR\_TYPE\_NOT\_-  
SUPPORTED, 684
  - LBMSDM\_ERR\_UNICODE\_-  
CONVERSION, 684
  - lbmsdm\_errmsg, 684
  - lbmsdm\_errnum, 684
  - LBMSDM\_FAILURE, 683
  - LBMSDM\_FIELD\_IS\_NULL, 683
  - LBMSDM\_INSUFFICIENT\_-  
BUFFER\_LENGTH, 683
  - lbmsdm\_iter\_create, 684
  - lbmsdm\_iter\_del, 685
  - lbmsdm\_iter\_del\_elem, 685
  - lbmsdm\_iter\_destroy, 685
  - lbmsdm\_iter\_first, 685
  - lbmsdm\_iter\_get\_element, 686
  - lbmsdm\_iter\_get\_elemlen, 686
  - lbmsdm\_iter\_get\_len, 686
  - lbmsdm\_iter\_get\_name, 687
  - lbmsdm\_iter\_get\_type, 687
  - lbmsdm\_iter\_is\_null, 687
  - lbmsdm\_iter\_next, 687
  - lbmsdm\_iter\_set\_null, 688
  - lbmsdm\_msg\_attr\_create, 688
  - lbmsdm\_msg\_attr\_delete, 688
  - lbmsdm\_msg\_attr\_dup, 688
  - lbmsdm\_msg\_attr\_getopt, 689
  - lbmsdm\_msg\_attr\_setopt, 689
  - lbmsdm\_msg\_attr\_str\_getopt, 690
  - lbmsdm\_msg\_attr\_str\_setopt, 690
  - lbmsdm\_msg\_clear, 690
  - lbmsdm\_msg\_clone, 691

- lbmsdm\_msg\_create, 691
- lbmsdm\_msg\_create\_ex, 691
- lbmsdm\_msg\_del\_elem\_idx, 692
- lbmsdm\_msg\_del\_elem\_name, 692
- lbmsdm\_msg\_del\_idx, 692
- lbmsdm\_msg\_del\_name, 692
- lbmsdm\_msg\_destroy, 693
- lbmsdm\_msg\_dump, 693
- lbmsdm\_msg\_get\_data, 693
- lbmsdm\_msg\_get\_data\_len, 694
- lbmsdm\_msg\_get\_element\_idx, 694
- lbmsdm\_msg\_get\_element\_name, 694
- lbmsdm\_msg\_get\_element\_idx, 695
- lbmsdm\_msg\_get\_element\_name, 695
- lbmsdm\_msg\_get\_field, 695
- lbmsdm\_msg\_get\_idx\_name, 696
- lbmsdm\_msg\_get\_len\_idx, 696
- lbmsdm\_msg\_get\_len\_name, 696
- lbmsdm\_msg\_get\_name\_idx, 697
- lbmsdm\_msg\_get\_type\_idx, 697
- lbmsdm\_msg\_get\_type\_name, 697
- lbmsdm\_msg\_is\_null\_idx, 698
- lbmsdm\_msg\_is\_null\_name, 698
- lbmsdm\_msg\_parse, 698
- lbmsdm\_msg\_parse\_ex, 699
- lbmsdm\_msg\_parse\_reuse, 699
- lbmsdm\_msg\_set\_null\_idx, 699
- lbmsdm\_msg\_set\_null\_name, 700
- LBMSDM\_NO\_MORE\_FIELDS, 683
- LBMSDM\_SUCCESS, 683
- LBMSDM\_TYPE\_ARRAY\_BLOB, 683
- LBMSDM\_TYPE\_ARRAY\_BOOLEAN, 682
- LBMSDM\_TYPE\_ARRAY\_DECIMAL, 682
- LBMSDM\_TYPE\_ARRAY\_DOUBLE, 682
- LBMSDM\_TYPE\_ARRAY\_FLOAT, 682
- LBMSDM\_TYPE\_ARRAY\_INT16, 682
- LBMSDM\_TYPE\_ARRAY\_INT32, 682
- LBMSDM\_TYPE\_ARRAY\_INT64, 682
- LBMSDM\_TYPE\_ARRAY\_INT8, 681
- LBMSDM\_TYPE\_ARRAY\_INVALID, 681
- LBMSDM\_TYPE\_ARRAY\_MESSAGE, 682
- LBMSDM\_TYPE\_ARRAY\_STRING, 682
- LBMSDM\_TYPE\_ARRAY\_TIMESTAMP, 682
- LBMSDM\_TYPE\_ARRAY\_UINT16, 681
- LBMSDM\_TYPE\_ARRAY\_UINT32, 682
- LBMSDM\_TYPE\_ARRAY\_UINT64, 682
- LBMSDM\_TYPE\_ARRAY\_UINT8, 681
- LBMSDM\_TYPE\_ARRAY\_UNICODE, 682
- lbmsdm\_win32\_static\_init, 700
- lbmsdm\_decimal\_t
  - lbmsdm.h, 681
- lbmsdm\_decimal\_t\_stct, 305
  - exp, 305
- LBMSDM\_TYPE\_ARRAY\_INT32, 682
- LBMSDM\_TYPE\_ARRAY\_INT64, 682
- LBMSDM\_TYPE\_ARRAY\_INT8, 682
- LBMSDM\_TYPE\_ARRAY\_MESSAGE, 683
- LBMSDM\_TYPE\_ARRAY\_STRING, 683
- LBMSDM\_TYPE\_ARRAY\_TIMESTAMP, 682
- LBMSDM\_TYPE\_ARRAY\_UINT16, 682
- LBMSDM\_TYPE\_ARRAY\_UINT32, 682
- LBMSDM\_TYPE\_ARRAY\_UINT64, 682
- LBMSDM\_TYPE\_ARRAY\_UINT8, 682
- LBMSDM\_TYPE\_ARRAY\_UNICODE, 683
- LBMSDM\_TYPE\_BLOB, 682
- LBMSDM\_TYPE\_BOOLEAN, 681
- LBMSDM\_TYPE\_DECIMAL, 682
- LBMSDM\_TYPE\_DOUBLE, 682
- LBMSDM\_TYPE\_FLOAT, 682
- LBMSDM\_TYPE\_INT16, 681
- LBMSDM\_TYPE\_INT32, 682
- LBMSDM\_TYPE\_INT64, 682
- LBMSDM\_TYPE\_INT8, 681
- LBMSDM\_TYPE\_INVALID, 681
- LBMSDM\_TYPE\_MESSAGE, 682
- LBMSDM\_TYPE\_STRING, 682
- LBMSDM\_TYPE\_TIMESTAMP, 682

- mant, [305](#)
- LBMSDM\_ERR\_ADDING\_FIELD
  - [lbmsdm.h](#), [684](#)
- LBMSDM\_ERR\_BAD\_TYPE
  - [lbmsdm.h](#), [683](#)
- LBMSDM\_ERR\_CANNOT\_CONVERT
  - [lbmsdm.h](#), [683](#)
- LBMSDM\_ERR\_DELETING\_FIELD
  - [lbmsdm.h](#), [684](#)
- LBMSDM\_ERR\_DUPLICATE\_FIELD
  - [lbmsdm.h](#), [683](#)
- LBMSDM\_ERR\_EINVAL
  - [lbmsdm.h](#), [683](#)
- LBMSDM\_ERR\_ELEMENT\_NOT\_FOUND
  - [lbmsdm.h](#), [684](#)
- LBMSDM\_ERR\_ENOMEM
  - [lbmsdm.h](#), [683](#)
- LBMSDM\_ERR\_FIELD\_IS\_NULL
  - [lbmsdm.h](#), [684](#)
- LBMSDM\_ERR\_FIELD\_NOT\_FOUND
  - [lbmsdm.h](#), [683](#)
- LBMSDM\_ERR\_INVALID\_FIELD\_NAME
  - [lbmsdm.h](#), [684](#)
- LBMSDM\_ERR\_ITERATOR\_INVALID
  - [lbmsdm.h](#), [684](#)
- LBMSDM\_ERR\_MSG\_INVALID
  - [lbmsdm.h](#), [683](#)
- LBMSDM\_ERR\_NAMETOOLONG
  - [lbmsdm.h](#), [683](#)
- LBMSDM\_ERR\_NOT\_ARRAY
  - [lbmsdm.h](#), [683](#)
- LBMSDM\_ERR\_NOT\_SCALAR
  - [lbmsdm.h](#), [684](#)
- LBMSDM\_ERR\_TYPE\_MISMATCH
  - [lbmsdm.h](#), [684](#)
- LBMSDM\_ERR\_TYPE\_NOT\_SUPPORTED
  - [lbmsdm.h](#), [684](#)
- LBMSDM\_ERR\_UNICODE\_CONVERSION
  - [lbmsdm.h](#), [684](#)
- [lbmsdm\\_errmsg](#)
  - [lbmsdm.h](#), [684](#)
- [lbmsdm\\_errnum](#)
  - [lbmsdm.h](#), [685](#)
- [lbmsdm.h](#), [684](#)
- LBMSDM\_FAILURE
  - [lbmsdm.h](#), [683](#)
- LBMSDM\_FIELD\_IS\_NULL
  - [lbmsdm.h](#), [683](#)
- LBMSDM\_INSUFFICIENT\_BUFFER\_LENGTH
  - [lbmsdm.h](#), [683](#)
- [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](#), [684](#)
- [lbmsdm\\_iter\\_del](#)
  - [lbmsdm.h](#), [685](#)

- lbmsdm\_iter\_del\_elem
  - lbmsdm.h, [685](#)
- lbmsdm\_iter\_destroy
  - lbmsdm.h, [685](#)
- lbmsdm\_iter\_first
  - lbmsdm.h, [685](#)
- 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\_element
  - lbmsdm.h, [686](#)
- lbmsdm\_iter\_get\_elementlen
  - lbmsdm.h, [686](#)
- 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, [686](#)
- lbmsdm\_iter\_get\_message
  - get\_scalar\_iter, [51](#)
- lbmsdm\_iter\_get\_message\_elem
  - get\_elem\_iter, [67](#)
- lbmsdm\_iter\_get\_name
  - lbmsdm.h, [687](#)
- 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, [687](#)
- 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, [687](#)
- lbmsdm\_iter\_next
  - lbmsdm.h, [687](#)
- 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, [688](#)
- 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, [688](#)
- lbmsdm\_msg\_attr\_delete
  - lbmsdm.h, [688](#)
- lbmsdm\_msg\_attr\_dup
  - lbmsdm.h, [688](#)
- lbmsdm\_msg\_attr\_getopt
  - lbmsdm.h, [689](#)
- lbmsdm\_msg\_attr\_setopt
  - lbmsdm.h, [689](#)
- lbmsdm\_msg\_attr\_str\_getopt
  - lbmsdm.h, [690](#)
- lbmsdm\_msg\_attr\_str\_setopt
  - lbmsdm.h, [690](#)
- lbmsdm\_msg\_clear
  - lbmsdm.h, [690](#)
- lbmsdm\_msg\_clone
  - lbmsdm.h, [691](#)
- lbmsdm\_msg\_create
  - lbmsdm.h, [691](#)
- lbmsdm\_msg\_create\_ex
  - lbmsdm.h, [691](#)
- lbmsdm\_msg\_del\_elem\_idx
  - lbmsdm.h, [692](#)
- lbmsdm\_msg\_del\_elem\_name
  - lbmsdm.h, [692](#)
- lbmsdm\_msg\_del\_idx
  - lbmsdm.h, [692](#)
- lbmsdm\_msg\_del\_name
  - lbmsdm.h, [692](#)
- lbmsdm\_msg\_destroy
  - lbmsdm.h, [693](#)
- lbmsdm\_msg\_dump
  - lbmsdm.h, [693](#)
- 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, [693](#)
- lbmsdm\_msg\_get\_data\_len
  - lbmsdm.h, [694](#)
- 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, [694](#)
- lbmsdm\_msg\_get\_element\_name
  - lbmsdm.h, [694](#)
- lbmsdm\_msg\_get\_elem\_len\_idx
  - lbmsdm.h, [695](#)
- lbmsdm\_msg\_get\_elem\_len\_name
  - lbmsdm.h, [695](#)
- lbmsdm\_msg\_get\_fldcnt
  - lbmsdm.h, [695](#)
- 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, [696](#)
- 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, [696](#)
- lbmsdm\_msg\_get\_len\_name
  - lbmsdm.h, [696](#)
- 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, [697](#)
- 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, [697](#)
- lbmsdm\_msg\_get\_type\_name
  - lbmsdm.h, [697](#)
- 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
  - get\_elem\_name, [63](#)
- 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, [698](#)
- lbmsdm\_msg\_is\_null\_name
  - lbmsdm.h, [698](#)
- lbmsdm\_msg\_parse
  - lbmsdm.h, [698](#)
- lbmsdm\_msg\_parse\_ex
  - lbmsdm.h, [699](#)
- lbmsdm\_msg\_parse\_reuse
  - lbmsdm.h, [699](#)
- 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, <a href="#">98</a>	lbmsdm_msg_set_int16_array_name set_array_name, <a href="#">90</a>
lbmsdm_msg_set_boolean_elem_name set_elem_name, <a href="#">103</a>	lbmsdm_msg_set_int16_elem_idx set_elem_idx, <a href="#">99</a>
lbmsdm_msg_set_boolean_idx set_idx, <a href="#">71</a>	lbmsdm_msg_set_int16_elem_name set_elem_name, <a href="#">104</a>
lbmsdm_msg_set_boolean_name set_name, <a href="#">76</a>	lbmsdm_msg_set_int16_idx set_idx, <a href="#">72</a>
lbmsdm_msg_set_decimal_array_idx set_array_idx, <a href="#">86</a>	lbmsdm_msg_set_int16_name set_name, <a href="#">77</a>
lbmsdm_msg_set_decimal_array_name set_array_name, <a href="#">90</a>	lbmsdm_msg_set_int32_array_idx set_array_idx, <a href="#">87</a>
lbmsdm_msg_set_decimal_elem_idx set_elem_idx, <a href="#">98</a>	lbmsdm_msg_set_int32_array_name set_array_name, <a href="#">91</a>
lbmsdm_msg_set_decimal_elem_name set_elem_name, <a href="#">103</a>	lbmsdm_msg_set_int32_elem_idx set_elem_idx, <a href="#">99</a>
lbmsdm_msg_set_decimal_idx set_idx, <a href="#">71</a>	lbmsdm_msg_set_int32_elem_name set_elem_name, <a href="#">104</a>
lbmsdm_msg_set_decimal_name set_name, <a href="#">76</a>	lbmsdm_msg_set_int32_idx set_idx, <a href="#">72</a>
lbmsdm_msg_set_double_array_idx set_array_idx, <a href="#">86</a>	lbmsdm_msg_set_int32_name set_name, <a href="#">77</a>
lbmsdm_msg_set_double_array_name set_array_name, <a href="#">90</a>	lbmsdm_msg_set_int64_array_idx set_array_idx, <a href="#">87</a>
lbmsdm_msg_set_double_elem_idx set_elem_idx, <a href="#">98</a>	lbmsdm_msg_set_int64_array_name set_array_name, <a href="#">91</a>
lbmsdm_msg_set_double_elem_name set_elem_name, <a href="#">104</a>	lbmsdm_msg_set_int64_elem_idx set_elem_idx, <a href="#">99</a>
lbmsdm_msg_set_double_idx set_idx, <a href="#">71</a>	lbmsdm_msg_set_int64_elem_name set_elem_name, <a href="#">104</a>
lbmsdm_msg_set_double_name set_name, <a href="#">76</a>	lbmsdm_msg_set_int64_idx set_idx, <a href="#">72</a>
lbmsdm_msg_set_float_array_idx set_array_idx, <a href="#">86</a>	lbmsdm_msg_set_int64_name set_name, <a href="#">77</a>
lbmsdm_msg_set_float_array_name set_array_name, <a href="#">90</a>	lbmsdm_msg_set_int8_array_idx set_array_idx, <a href="#">87</a>
lbmsdm_msg_set_float_elem_idx set_elem_idx, <a href="#">99</a>	lbmsdm_msg_set_int8_array_name set_array_name, <a href="#">91</a>
lbmsdm_msg_set_float_elem_name set_elem_name, <a href="#">104</a>	lbmsdm_msg_set_int8_elem_idx set_elem_idx, <a href="#">99</a>
lbmsdm_msg_set_float_idx set_idx, <a href="#">72</a>	lbmsdm_msg_set_int8_elem_name set_elem_name, <a href="#">104</a>
lbmsdm_msg_set_float_name set_name, <a href="#">77</a>	lbmsdm_msg_set_int8_idx set_idx, <a href="#">72</a>
lbmsdm_msg_set_int16_array_idx set_array_idx, <a href="#">86</a>	lbmsdm_msg_set_int8_name set_name, <a href="#">77</a>

- 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, [699](#)
- lbmsdm\_msg\_set\_null\_name  
lbmsdm.h, [700](#)
- 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, [683](#)
- LBMSDM\_SUCCESS
  - lbmsdm.h, [683](#)
- LBMSDM\_TYPE\_ARRAY\_BLOB
  - lbmsdm.h, [683](#)
- LBMSDM\_TYPE\_ARRAY\_BOOLEAN
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_ARRAY\_DECIMAL
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_ARRAY\_DOUBLE
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_ARRAY\_FLOAT
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_ARRAY\_INT16
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_ARRAY\_INT32
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_ARRAY\_INT64
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_ARRAY\_INT8
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_ARRAY\_MESSAGE
  - lbmsdm.h, [683](#)
- LBMSDM\_TYPE\_ARRAY\_STRING
  - lbmsdm.h, [683](#)
- LBMSDM\_TYPE\_ARRAY\_-  
TIMESTAMP
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_ARRAY\_UINT16
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_ARRAY\_UINT32
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_ARRAY\_UINT64
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_ARRAY\_UINT8
  - lbmsdm.h, [682](#)
- lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_ARRAY\_UNICODE
  - lbmsdm.h, [683](#)
- LBMSDM\_TYPE\_BLOB
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_BOOLEAN
  - lbmsdm.h, [681](#)
- LBMSDM\_TYPE\_DECIMAL
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_DOUBLE
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_FLOAT
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_INT16
  - lbmsdm.h, [681](#)
- LBMSDM\_TYPE\_INT32
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_INT64
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_INT8
  - lbmsdm.h, [681](#)
- LBMSDM\_TYPE\_INVALID
  - lbmsdm.h, [681](#)
- LBMSDM\_TYPE\_MESSAGE
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_STRING
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_TIMESTAMP
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_UINT16
  - lbmsdm.h, [681](#)
- LBMSDM\_TYPE\_UINT32
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_UINT64
  - lbmsdm.h, [682](#)
- LBMSDM\_TYPE\_UINT8
  - lbmsdm.h, [681](#)
- LBMSDM\_TYPE\_UNICODE
  - lbmsdm.h, [682](#)
- lbmsdm\_win32\_static\_init
  - lbmsdm.h, [700](#)
- lbtipec
  - lbm\_rcv\_transport\_stats\_t\_stct, [197](#)
  - lbm\_src\_transport\_stats\_t\_stct, [245](#)
- lbtrdma
  - lbm\_rcv\_transport\_stats\_t\_stct, [197](#)

- lbm\_src\_transport\_stats\_t\_stct, 245
- lbtrm
  - lbm\_rcv\_transport\_stats\_t\_stct, 197
  - lbm\_src\_transport\_stats\_t\_stct, 245
- lbtrm\_unknown\_msgs\_rcved
  - lbm\_context\_stats\_t\_stct, 126
- lbtru
  - lbm\_rcv\_transport\_stats\_t\_stct, 197
  - lbm\_src\_transport\_stats\_t\_stct, 245
- lbtru\_unknown\_msgs\_rcved
  - lbm\_context\_stats\_t\_stct, 126
- lbtsmx
  - lbm\_rcv\_transport\_stats\_t\_stct, 197
  - lbm\_src\_transport\_stats\_t\_stct, 245
- len
  - lbm\_apphdr\_chain\_elem\_t\_stct, 113
  - lbm\_msg\_t\_stct, 156
  - lbmpdm\_field\_value\_stct\_t, 302
- len\_arr
  - lbmpdm\_field\_value\_stct\_t, 302
- lost
  - lbm\_rcv\_transport\_stats\_lbtrm\_t\_stct, 183
  - lbm\_rcv\_transport\_stats\_lbtru\_t\_stct, 190
- low\_rxreq\_max\_sequence\_number
  - lbm\_ume\_rcv\_recovery\_info\_ex\_func\_info\_t\_stct, 256
- low\_sequence\_number
  - lbm\_ume\_rcv\_recovery\_info\_ex\_func\_info\_t\_stct, 257
- lu
  - lbm\_umq\_ulb\_application\_set\_attr\_t\_stct, 278
  - lbm\_umq\_ulb\_receiver\_type\_attr\_t\_stct, 279
- mant
  - lbmpdm\_decimal\_t, 300
  - lbmsdm\_decimal\_t\_stct, 305
- mAttributeBlockLength
  - lbmmon\_packet\_hdr\_t\_stct, 292
- mc\_group
  - lbm\_transport\_source\_info\_t\_stct, 251
- mCtxDeserialize
  - lbmmon\_format\_func\_t\_stct, 289
- mCtxSerialize
  - lbmmon\_format\_func\_t\_stct, 289
- mDataLength
  - lbmmon\_packet\_hdr\_t\_stct, 292
- mEntryCount
  - lbmmon\_attr\_block\_t\_stct, 285
- mEntryLength
  - lbmmon\_attr\_block\_t\_stct, 285
- mErrorMessage
  - lbmmon\_format\_func\_t\_stct, 289
  - lbmmon\_transport\_func\_t\_stct, 297
- messages
  - lbm\_flight\_size\_inflight\_t\_stct, 146
- mEvqDeserialize
  - lbmmon\_format\_func\_t\_stct, 289
- mEvqSerialize
  - lbmmon\_format\_func\_t\_stct, 290
- mFiller
  - lbmmon\_packet\_hdr\_t\_stct, 292
- mFinish
  - lbmmon\_format\_func\_t\_stct, 290
- mFinishReceiver
  - lbmmon\_transport\_func\_t\_stct, 297
- mFinishSource
  - lbmmon\_transport\_func\_t\_stct, 297
- mInit
  - lbmmon\_format\_func\_t\_stct, 290
- mInitReceiver
  - lbmmon\_transport\_func\_t\_stct, 297
- mInitSource
  - lbmmon\_transport\_func\_t\_stct, 297
- mLength
  - lbmmon\_attr\_entry\_t\_stct, 286
- mRcvDeserialize
  - lbmmon\_format\_func\_t\_stct, 290
- mRcvSerialize
  - lbmmon\_format\_func\_t\_stct, 290
- mRcvTopicDeserialize
  - lbmmon\_format\_func\_t\_stct, 290
- mRcvTopicSerialize
  - lbmmon\_format\_func\_t\_stct, 290
- mReceive
  - lbmmon\_transport\_func\_t\_stct, 297
- mSend
  - lbmmon\_transport\_func\_t\_stct, 298

- msg
  - lbm\_umq\_queue\_msg\_status\_t, 274
- msg\_clientd
  - lbm\_src\_event\_sequence\_number\_info\_t\_stct, 209
  - lbm\_src\_event\_ume\_ack\_ex\_info\_t\_stct, 211
  - lbm\_src\_event\_ume\_ack\_info\_t\_stct, 213
  - lbm\_src\_event\_umq\_message\_id\_info\_t\_stct, 220
  - lbm\_src\_event\_umq\_stability\_ack\_info\_ex\_t\_stct, 224
  - lbm\_src\_event\_umq\_ulb\_message\_info\_ex\_t\_stct, 226
- msg\_id
  - lbm\_src\_event\_umq\_message\_id\_info\_t\_stct, 220
  - lbm\_src\_event\_umq\_stability\_ack\_info\_ex\_t\_stct, 224
  - lbm\_src\_event\_umq\_ulb\_message\_info\_ex\_t\_stct, 226
- msgid
  - lbm\_umq\_queue\_msg\_status\_t, 274
- msgs
  - lbm\_rcv\_umq\_queue\_msg\_list\_info\_t, 201
  - lbm\_rcv\_umq\_queue\_msg\_retrieve\_info\_t, 202
- msgs\_rcved
  - lbm\_rcv\_transport\_stats\_lbtpic\_t\_stct, 178
  - lbm\_rcv\_transport\_stats\_lbtrdma\_t\_stct, 180
  - lbm\_rcv\_transport\_stats\_lbtrm\_t\_stct, 183
  - lbm\_rcv\_transport\_stats\_lbtru\_t\_stct, 190
  - lbm\_rcv\_transport\_stats\_lbtsmx\_t\_stct, 194
- msgs\_sent
  - lbm\_src\_transport\_stats\_lbtpic\_t\_stct, 234
  - lbm\_src\_transport\_stats\_lbtrdma\_t\_stct, 235
  - lbm\_src\_transport\_stats\_lbtrm\_t\_stct, 236
  - lbm\_src\_transport\_stats\_lbtru\_t\_stct, 240
  - lbm\_src\_transport\_stats\_lbtsmx\_t\_stct, 243
- mSignature
  - lbmmon\_packet\_hdr\_t\_stct, 292
- mSrcDeserialize
  - lbmmon\_format\_func\_t\_stct, 290
- mSrcSerialize
  - lbmmon\_format\_func\_t\_stct, 291
- mType
  - lbmmon\_attr\_entry\_t\_stct, 286
  - lbmmon\_packet\_hdr\_t\_stct, 292
- mWildcardRcvDeserialize
  - lbmmon\_format\_func\_t\_stct, 291
- mWildcardRcvSerialize
  - lbmmon\_format\_func\_t\_stct, 291
- nak\_pkts\_rcved
  - lbm\_src\_transport\_stats\_lbtrm\_t\_stct, 236
  - lbm\_src\_transport\_stats\_lbtru\_t\_stct, 240
- nak\_pkts\_sent
  - lbm\_rcv\_transport\_stats\_lbtrm\_t\_stct, 183
  - lbm\_rcv\_transport\_stats\_lbtru\_t\_stct, 190
- nak\_stm\_max
  - lbm\_rcv\_transport\_stats\_lbtrm\_t\_stct, 184
  - lbm\_rcv\_transport\_stats\_lbtru\_t\_stct, 190
- nak\_stm\_mean
  - lbm\_rcv\_transport\_stats\_lbtrm\_t\_stct, 184
  - lbm\_rcv\_transport\_stats\_lbtru\_t\_stct, 191
- nak\_stm\_min
  - lbm\_rcv\_transport\_stats\_lbtrm\_t\_stct, 184
  - lbm\_rcv\_transport\_stats\_lbtru\_t\_stct, 191
- nak\_tx\_max

- lbm\_rcv\_transport\_stats\_lbtrm\_t - stct, [184](#)
- lbm\_rcv\_transport\_stats\_lbtru\_t - stct, [191](#)
- nak\_tx\_mean
  - lbm\_rcv\_transport\_stats\_lbtrm\_t - stct, [184](#)
  - lbm\_rcv\_transport\_stats\_lbtru\_t - stct, [191](#)
- nak\_tx\_min
  - lbm\_rcv\_transport\_stats\_lbtrm\_t - stct, [184](#)
  - lbm\_rcv\_transport\_stats\_lbtru\_t - stct, [191](#)
- naks\_ignored
  - lbm\_src\_transport\_stats\_lbtrm\_t - stct, [236](#)
  - lbm\_src\_transport\_stats\_lbtru\_t - stct, [240](#)
- naks\_rcved
  - lbm\_src\_transport\_stats\_lbtrm\_t - stct, [237](#)
  - lbm\_src\_transport\_stats\_lbtru\_t - stct, [241](#)
- naks\_rx\_delay\_ignored
  - lbm\_src\_transport\_stats\_lbtrm\_t - stct, [237](#)
  - lbm\_src\_transport\_stats\_lbtru\_t - stct, [241](#)
- naks\_sent
  - lbm\_rcv\_transport\_stats\_lbtrm\_t - stct, [185](#)
  - lbm\_rcv\_transport\_stats\_lbtru\_t - stct, [191](#)
- naks\_shed
  - lbm\_src\_transport\_stats\_lbtrm\_t - stct, [237](#)
  - lbm\_src\_transport\_stats\_lbtru\_t - stct, [241](#)
- name
  - lbm\_ume\_store\_name\_entry\_t\_stct, [267](#)
  - lbm\_umq\_queue\_entry\_t\_stct, [273](#)
- ncfs\_ignored
  - lbm\_rcv\_transport\_stats\_lbtrm\_t - stct, [185](#)
  - lbm\_rcv\_transport\_stats\_lbtru\_t - stct, [192](#)
- ncfs\_rx\_delay
  - lbm\_rcv\_transport\_stats\_lbtrm\_t - stct, [185](#)
  - lbm\_rcv\_transport\_stats\_lbtru\_t - stct, [192](#)
- ncfs\_shed
  - lbm\_rcv\_transport\_stats\_lbtrm\_t - stct, [185](#)
  - lbm\_rcv\_transport\_stats\_lbtru\_t - stct, [192](#)
- ncfs\_unknown
  - lbm\_rcv\_transport\_stats\_lbtrm\_t - stct, [186](#)
  - lbm\_rcv\_transport\_stats\_lbtru\_t - stct, [192](#)
- num\_appsets
  - lbm\_umq\_queue\_topic\_t\_stct, [277](#)
- num\_arr\_elem
  - lbmpdm\_field\_info\_attr\_stct\_t, [301](#)
  - lbmpdm\_field\_value\_stct\_t, [302](#)
- num\_clients
  - lbm\_src\_transport\_stats\_lbtrm\_t - stct, [234](#)
  - lbm\_src\_transport\_stats\_lbtrdma\_t\_stct, [235](#)
  - lbm\_src\_transport\_stats\_lbtru\_t - stct, [241](#)
  - lbm\_src\_transport\_stats\_lbtrmx\_t - stct, [243](#)
  - lbm\_src\_transport\_stats\_tcp\_t\_stct, [247](#)
- num\_msgs
  - lbm\_rcv\_umq\_queue\_msg\_list\_info\_t, [201](#)
  - lbm\_rcv\_umq\_queue\_msg\_retrieve\_info\_t, [202](#)
- num\_topics
  - lbm\_ctx\_umq\_queue\_topic\_list\_info\_t, [131](#)
- offset
  - lbm\_msg\_fragment\_info\_t\_stct, [152](#)
- otid
  - lbm\_rcv\_topic\_stats\_t\_stct, [174](#)

---

out\_of\_order  
     lbm\_rcv\_transport\_stats\_lbtrm\_t\_-  
         stct, 186

password  
     lbm\_umm\_info\_t\_stct, 268

pattern  
     lbm\_wildcard\_rcv\_stats\_t\_stct, 284

PDM\_DEFN\_INT\_FIELD\_NAMES  
     lbmpdm.h, 620

PDM\_DEFN\_STR\_FIELD\_NAMES  
     lbmpdm.h, 620

PDM\_ERR\_CREATE\_BUFFER  
     lbmpdm.h, 620

PDM\_ERR\_CREATE\_SECTION  
     lbmpdm.h, 621

PDM\_ERR\_DEFN\_INVALID  
     lbmpdm.h, 621

PDM\_ERR\_EINVAL  
     lbmpdm.h, 621

PDM\_ERR\_FIELD\_IS\_NULL  
     lbmpdm.h, 621

PDM\_ERR\_FIELD\_NOT\_FOUND  
     lbmpdm.h, 621

PDM\_ERR\_INSUFFICIENT\_-  
     BUFFER\_LENGTH  
     lbmpdm.h, 621

PDM\_ERR\_MSG\_INVALID  
     lbmpdm.h, 621

PDM\_ERR\_NO\_MORE\_FIELDS  
     lbmpdm.h, 621

PDM\_ERR\_NOMEM  
     lbmpdm.h, 621

PDM\_ERR\_REQ\_FIELD\_NOT\_SET  
     lbmpdm.h, 621

PDM\_FAILURE  
     lbmpdm.h, 622

PDM\_FALSE  
     lbmpdm.h, 622

PDM\_FIELD\_INFO\_FLAG\_FIXED\_-  
     STR\_LEN  
     lbmpdm.h, 622

PDM\_FIELD\_INFO\_FLAG\_NUM\_-  
     ARR\_ELEM  
     lbmpdm.h, 622

PDM\_FIELD\_INFO\_FLAG\_REQ  
     lbmpdm.h, 622

PDM\_INTERNAL\_TYPE\_INVALID  
     lbmpdm.h, 622

PDM\_ITER\_INVALID\_FIELD\_-  
     HANDLE  
     lbmpdm.h, 622

PDM\_MSG\_FLAG\_DEL\_DEFN\_-  
     WHEN\_REPLACED  
     lbmpdm.h, 622

PDM\_MSG\_FLAG\_INCL\_DEFN  
     lbmpdm.h, 622

PDM\_MSG\_FLAG\_NEED\_BYTE\_-  
     SWAP  
     lbmpdm.h, 623

PDM\_MSG\_FLAG\_TRY\_LOAD\_-  
     DEFN\_FROM\_CACHE  
     lbmpdm.h, 623

PDM\_MSG\_FLAG\_USE\_MSG\_-  
     DEFN\_IF\_NEEDED  
     lbmpdm.h, 623

PDM\_MSG\_FLAG\_VAR\_OR\_OPT\_-  
     FLDS\_SET  
     lbmpdm.h, 623

PDM\_MSG\_VER\_POLICY\_BEST  
     lbmpdm.h, 623

PDM\_MSG\_VER\_POLICY\_EXACT  
     lbmpdm.h, 623

PDM\_SUCCESS  
     lbmpdm.h, 623

PDM\_TRUE  
     lbmpdm.h, 623

PDM\_TYPE\_BLOB  
     lbmpdm.h, 623

PDM\_TYPE\_BLOB\_ARR  
     lbmpdm.h, 624

PDM\_TYPE\_BOOLEAN  
     lbmpdm.h, 624

PDM\_TYPE\_BOOLEAN\_ARR  
     lbmpdm.h, 624

PDM\_TYPE\_DECIMAL  
     lbmpdm.h, 624

PDM\_TYPE\_DECIMAL\_ARR  
     lbmpdm.h, 624

PDM\_TYPE\_DOUBLE  
     lbmpdm.h, 624

PDM\_TYPE\_DOUBLE\_ARR

- lbmpdm.h, [624](#)
- PDM\_TYPE\_FIX\_STRING
  - lbmpdm.h, [624](#)
- PDM\_TYPE\_FIX\_STRING\_ARR
  - lbmpdm.h, [624](#)
- PDM\_TYPE\_FIX\_UNICODE
  - lbmpdm.h, [624](#)
- PDM\_TYPE\_FIX\_UNICODE\_ARR
  - lbmpdm.h, [625](#)
- PDM\_TYPE\_FLOAT
  - lbmpdm.h, [625](#)
- PDM\_TYPE\_FLOAT\_ARR
  - lbmpdm.h, [625](#)
- PDM\_TYPE\_INT16
  - lbmpdm.h, [625](#)
- PDM\_TYPE\_INT16\_ARR
  - lbmpdm.h, [625](#)
- PDM\_TYPE\_INT32
  - lbmpdm.h, [625](#)
- PDM\_TYPE\_INT32\_ARR
  - lbmpdm.h, [625](#)
- PDM\_TYPE\_INT64
  - lbmpdm.h, [625](#)
- PDM\_TYPE\_INT64\_ARR
  - lbmpdm.h, [625](#)
- PDM\_TYPE\_INT8
  - lbmpdm.h, [625](#)
- PDM\_TYPE\_INT8\_ARR
  - lbmpdm.h, [626](#)
- PDM\_TYPE\_MESSAGE
  - lbmpdm.h, [626](#)
- PDM\_TYPE\_MESSAGE\_ARR
  - lbmpdm.h, [626](#)
- PDM\_TYPE\_STRING
  - lbmpdm.h, [626](#)
- PDM\_TYPE\_STRING\_ARR
  - lbmpdm.h, [626](#)
- PDM\_TYPE\_TIMESTAMP
  - lbmpdm.h, [626](#)
- PDM\_TYPE\_TIMESTAMP\_ARR
  - lbmpdm.h, [626](#)
- PDM\_TYPE\_UINT16
  - lbmpdm.h, [626](#)
- PDM\_TYPE\_UINT16\_ARR
  - lbmpdm.h, [626](#)
- PDM\_TYPE\_UINT32
  - lbmpdm.h, [626](#)
- PDM\_TYPE\_UINT32\_ARR
  - lbmpdm.h, [627](#)
- PDM\_TYPE\_UINT64
  - lbmpdm.h, [627](#)
- PDM\_TYPE\_UINT64\_ARR
  - lbmpdm.h, [627](#)
- PDM\_TYPE\_UINT8
  - lbmpdm.h, [627](#)
- PDM\_TYPE\_UINT8\_ARR
  - lbmpdm.h, [627](#)
- PDM\_TYPE\_UNICODE
  - lbmpdm.h, [627](#)
- PDM\_TYPE\_UNICODE\_ARR
  - lbmpdm.h, [627](#)
- properties
  - lbm\_msg\_t\_stct, [157](#)
  - lbm\_src\_send\_ex\_info\_t\_stct, [232](#)
- queue
  - lbm\_context\_event\_umq\_-
    - registration\_complete\_ex\_-
      - t\_stct, [119](#)
  - lbm\_context\_event\_umq\_-
    - registration\_ex\_t\_stct, [121](#)
  - lbm\_msg\_umq\_registration\_-
    - complete\_ex\_t\_stct, [171](#)
  - lbm\_src\_event\_umq\_registration\_-
    - complete\_ex\_t\_stct, [222](#)
  - lbm\_src\_event\_umq\_stability\_ack\_-
    - info\_ex\_t\_stct, [224](#)
- queue\_id
  - lbm\_context\_event\_umq\_-
    - registration\_complete\_ex\_-
      - t\_stct, [119](#)
  - lbm\_context\_event\_umq\_-
    - registration\_ex\_t\_stct, [121](#)
  - lbm\_msg\_umq\_deregistration\_-
    - complete\_ex\_t\_stct, [166](#)
  - lbm\_msg\_umq\_index\_assigned\_-
    - ex\_t\_stct, [167](#)
  - lbm\_msg\_umq\_index\_assignment\_-
    - eligibility\_start\_complete\_ex\_-
      - t\_stct, [168](#)
  - lbm\_msg\_umq\_index\_assignment\_-
    - eligibility\_stop\_complete\_ex\_-

- t\_stct, 169
- lbm\_msg\_umq\_index\_released\_ex\_t\_stct, 170
- lbm\_msg\_umq\_registration\_complete\_ex\_t\_stct, 171
- lbm\_src\_event\_umq\_registration\_complete\_ex\_t\_stct, 222
- lbm\_src\_event\_umq\_stability\_ack\_info\_ex\_t\_stct, 224
- queue\_instance
  - lbm\_context\_event\_umq\_registration\_ex\_t\_stct, 121
  - lbm\_src\_event\_umq\_stability\_ack\_info\_ex\_t\_stct, 224
- queue\_instance\_index
  - lbm\_context\_event\_umq\_registration\_ex\_t\_stct, 121
  - lbm\_src\_event\_umq\_stability\_ack\_info\_ex\_t\_stct, 224
- rctlr\_data\_msgs
  - lbm\_src\_transport\_stats\_lbtrm\_t\_stct, 237
- rctlr\_rx\_msgs
  - lbm\_src\_transport\_stats\_lbtrm\_t\_stct, 237
- rcv\_cb\_svc\_time\_max
  - lbm\_context\_stats\_t\_stct, 126
- rcv\_cb\_svc\_time\_mean
  - lbm\_context\_stats\_t\_stct, 127
- rcv\_cb\_svc\_time\_min
  - lbm\_context\_stats\_t\_stct, 127
- rcv\_registration\_id
  - lbm\_msg\_ume\_deregistration\_ex\_t\_stct, 160
  - lbm\_msg\_ume\_registration\_ex\_t\_stct, 163
  - lbm\_msg\_ume\_registration\_t\_stct, 165
  - lbm\_src\_event\_ume\_ack\_ex\_info\_t\_stct, 211
  - lbm\_src\_event\_ume\_ack\_info\_t\_stct, 213
- receiver
  - lbm\_src\_event\_umq\_ulb\_message\_info\_ex\_t\_stct, 226
  - lbm\_src\_event\_umq\_ulb\_receiver\_info\_ex\_t\_stct, 227
- regid
  - lbm\_umq\_msgid\_t\_stct, 272
  - lbm\_umq\_queue\_entry\_t\_stct, 273
- registration\_id
  - lbm\_context\_event\_umq\_registration\_complete\_ex\_t\_stct, 119
  - lbm\_context\_event\_umq\_registration\_ex\_t\_stct, 122
  - lbm\_src\_event\_ume\_deregistration\_ex\_t\_stct, 214
  - lbm\_src\_event\_ume\_registration\_ex\_t\_stct, 217
  - lbm\_src\_event\_ume\_registration\_t\_stct, 219
  - lbm\_src\_event\_umq\_ulb\_message\_info\_ex\_t\_stct, 226
  - lbm\_src\_event\_umq\_ulb\_receiver\_info\_ex\_t\_stct, 227
  - lbm\_ume\_store\_entry\_t\_stct, 264
  - lbm\_ume\_store\_name\_entry\_t\_stct, 267
- req
  - lbmpdm\_field\_info\_attr\_stct\_t, 301
- reserved
  - lbm\_umq\_queue\_topic\_t\_stct, 277
- reserved1
  - lbm\_rcv\_transport\_stats\_lbtsmx\_t\_stct, 194
- resolver\_ip
  - lbm\_ucast\_resolver\_entry\_t\_stct, 253
- resp\_blocked
  - lbm\_context\_stats\_t\_stct, 127
- resp\_msgs
  - lbm\_event\_queue\_stats\_t\_stct, 140
- resp\_msgs\_svc\_max
  - lbm\_event\_queue\_stats\_t\_stct, 140
- resp\_msgs\_svc\_mean
  - lbm\_event\_queue\_stats\_t\_stct, 140
- resp\_msgs\_svc\_min
  - lbm\_event\_queue\_stats\_t\_stct, 141
- resp\_msgs\_tot

- lbm\_event\_queue\_stats\_t\_stct, [141](#)
- resp\_would\_block
  - lbm\_context\_stats\_t\_stct, [127](#)
- response
  - lbm\_msg\_t\_stct, [157](#)
- rx\_bytes\_sent
  - lbm\_src\_transport\_stats\_lbtrm\_t\_stct, [238](#)
  - lbm\_src\_transport\_stats\_lbtru\_t\_stct, [241](#)
- rxs\_sent
  - lbm\_src\_transport\_stats\_lbtrm\_t\_stct, [238](#)
  - lbm\_src\_transport\_stats\_lbtru\_t\_stct, [242](#)
- send\_blocked
  - lbm\_context\_stats\_t\_stct, [128](#)
- send\_would\_block
  - lbm\_context\_stats\_t\_stct, [128](#)
- sequence\_number
  - lbm\_msg\_gateway\_info\_t\_stct, [153](#)
  - lbm\_msg\_t\_stct, [157](#)
  - lbm\_msg\_ume\_deregistration\_ex\_t\_stct, [160](#)
  - lbm\_msg\_ume\_registration\_complete\_ex\_t\_stct, [162](#)
  - lbm\_msg\_ume\_registration\_ex\_t\_stct, [163](#)
  - lbm\_src\_event\_ume\_ack\_ex\_info\_t\_stct, [211](#)
  - lbm\_src\_event\_ume\_ack\_info\_t\_stct, [213](#)
  - lbm\_src\_event\_ume\_deregistration\_ex\_t\_stct, [214](#)
  - lbm\_src\_event\_ume\_registration\_complete\_ex\_t\_stct, [216](#)
  - lbm\_src\_event\_ume\_registration\_ex\_t\_stct, [217](#)
- servers
  - lbm\_umm\_info\_t\_stct, [268](#)
- session\_id
  - lbm\_transport\_source\_info\_t\_stct, [251](#)
- Set a field value in a message by field index, [70](#)
- 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, [75](#)
- 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, [80](#)
- 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](#)
- set\_array\_idx
  - lbmsdm\_msg\_set\_blob\_array\_idx, [86](#)
  - lbmsdm\_msg\_set\_boolean\_array\_idx, [86](#)
  - lbmsdm\_msg\_set\_decimal\_array\_idx, [86](#)
  - lbmsdm\_msg\_set\_double\_array\_idx, [86](#)
  - lbmsdm\_msg\_set\_float\_array\_idx, [86](#)
  - lbmsdm\_msg\_set\_int16\_array\_idx, [86](#)
  - lbmsdm\_msg\_set\_int32\_array\_idx, [87](#)
  - lbmsdm\_msg\_set\_int64\_array\_idx, [87](#)
  - lbmsdm\_msg\_set\_int8\_array\_idx, [87](#)
  - lbmsdm\_msg\_set\_message\_array\_idx, [87](#)
  - lbmsdm\_msg\_set\_string\_array\_idx, [87](#)
  - lbmsdm\_msg\_set\_timestamp\_array\_idx, [87](#)
  - 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
  - umeblocks.h, 702
- source
  - lbm\_msg\_gateway\_info\_t\_stct, 153
  - lbm\_msg\_t\_stct, 157
  - lbm\_rcv\_topic\_stats\_t\_stct, 174
  - lbm\_rcv\_transport\_stats\_t\_stct, 197
  - lbm\_src\_transport\_stats\_t\_stct, 245
  - lbm\_ume\_rcv\_recovery\_info\_ex\_func\_info\_t\_stct, 257
  - lbm\_ume\_rcv\_regid\_ex\_func\_info\_t\_stct, 259
- source\_clientd
  - lbm\_msg\_t\_stct, 157
  - lbm\_ume\_rcv\_recovery\_info\_ex\_func\_info\_t\_stct, 257
  - lbm\_ume\_rcv\_regid\_ex\_func\_info\_t\_stct, 259
- source\_events
  - lbm\_event\_queue\_stats\_t\_stct, 141
- source\_events\_svc\_max
  - lbm\_event\_queue\_stats\_t\_stct, 141
- source\_events\_svc\_mean
  - lbm\_event\_queue\_stats\_t\_stct, 141
- source\_events\_svc\_min
  - lbm\_event\_queue\_stats\_t\_stct, 141
- source\_events\_tot
  - lbm\_event\_queue\_stats\_t\_stct, 142
- source\_port
  - lbm\_ucast\_resolver\_entry\_t\_stct, 253
- src\_ip
  - lbm\_transport\_source\_info\_t\_stct, 251
- src\_port
  - lbm\_transport\_source\_info\_t\_stct, 251
- src\_registration\_id
  - lbm\_msg\_ume\_deregistration\_ex\_t\_stct, 160
  - lbm\_msg\_ume\_registration\_ex\_t\_stct, 163
  - lbm\_msg\_ume\_registration\_t\_stct, 165
  - lbm\_ume\_rcv\_regid\_ex\_func\_info\_t\_stct, 259
- src\_session\_id
  - lbm\_msg\_ume\_registration\_complete\_ex\_t\_stct, 162
  - lbm\_msg\_ume\_registration\_ex\_t\_stct, 163
  - lbm\_ume\_rcv\_recovery\_info\_ex\_func\_info\_t\_stct, 257
- stamp
  - lbm\_umq\_msgid\_t\_stct, 272
- start\_sequence\_number
  - lbm\_msg\_fragment\_info\_t\_stct, 152
- state
  - lbm\_src\_event\_flight\_size\_notification\_t\_stct, 208
- status
  - lbm\_async\_operation\_info\_t, 117
  - lbm\_umq\_queue\_msg\_status\_t, 274
  - lbm\_umq\_queue\_topic\_status\_t, 276

- store
  - lbm\_msg\_ume\_deregistration\_ex\_t\_stct, 161
  - lbm\_msg\_ume\_registration\_ex\_t\_stct, 164
  - lbm\_src\_event\_ume\_ack\_ex\_info\_t\_stct, 211
  - lbm\_src\_event\_ume\_deregistration\_ex\_t\_stct, 214
  - lbm\_src\_event\_ume\_registration\_ex\_t\_stct, 217
  - lbm\_ume\_rcv\_regid\_ex\_func\_info\_t\_stct, 260
- store\_index
  - lbm\_msg\_ume\_deregistration\_ex\_t\_stct, 161
  - lbm\_msg\_ume\_registration\_ex\_t\_stct, 164
  - lbm\_src\_event\_ume\_ack\_ex\_info\_t\_stct, 212
  - lbm\_src\_event\_ume\_deregistration\_ex\_t\_stct, 215
  - lbm\_src\_event\_ume\_registration\_ex\_t\_stct, 217
  - lbm\_ume\_rcv\_regid\_ex\_func\_info\_t\_stct, 260
- subtype
  - lbm\_apphdr\_chain\_elem\_t\_stct, 113
- tcp
  - lbm\_rcv\_transport\_stats\_t\_stct, 197
  - lbm\_src\_transport\_stats\_t\_stct, 245
- tcp\_port
  - lbm\_ume\_store\_entry\_t\_stct, 264
- timer\_events
  - lbm\_event\_queue\_stats\_t\_stct, 142
- timer\_events\_svc\_max
  - lbm\_event\_queue\_stats\_t\_stct, 142
- timer\_events\_svc\_mean
  - lbm\_event\_queue\_stats\_t\_stct, 142
- timer\_events\_svc\_min
  - lbm\_event\_queue\_stats\_t\_stct, 142
- timer\_events\_tot
  - lbm\_event\_queue\_stats\_t\_stct, 142
- topic
  - lbm\_rcv\_topic\_stats\_t\_stct, 174
  - lbm\_umq\_queue\_topic\_status\_t, 276
- topic\_idx
  - lbm\_rcv\_topic\_stats\_t\_stct, 174
  - lbm\_transport\_source\_info\_t\_stct, 251
- topic\_name
  - lbm\_msg\_t\_stct, 157
  - lbm\_umq\_queue\_topic\_t\_stct, 277
- topicless\_im\_msgs
  - lbm\_event\_queue\_stats\_t\_stct, 143
- topicless\_im\_msgs\_svc\_max
  - lbm\_event\_queue\_stats\_t\_stct, 143
- topicless\_im\_msgs\_svc\_mean
  - lbm\_event\_queue\_stats\_t\_stct, 143
- topicless\_im\_msgs\_svc\_min
  - lbm\_event\_queue\_stats\_t\_stct, 143
- topicless\_im\_msgs\_tot
  - lbm\_event\_queue\_stats\_t\_stct, 143
- topics
  - lbm\_ctx\_umq\_queue\_topic\_list\_info\_t, 131
- total\_message\_length
  - lbm\_msg\_fragment\_info\_t\_stct, 152
- tr\_bytes\_rcved
  - lbm\_context\_stats\_t\_stct, 128
- tr\_bytes\_sent
  - lbm\_context\_stats\_t\_stct, 128
- tr\_dgrams\_dropped\_malformed
  - lbm\_context\_stats\_t\_stct, 128
- tr\_dgrams\_dropped\_type
  - lbm\_context\_stats\_t\_stct, 128
- tr\_dgrams\_dropped\_ver
  - lbm\_context\_stats\_t\_stct, 128
- tr\_dgrams\_rcved
  - lbm\_context\_stats\_t\_stct, 129
- tr\_dgrams\_send\_failed
  - lbm\_context\_stats\_t\_stct, 129
- tr\_dgrams\_sent
  - lbm\_context\_stats\_t\_stct, 129
- tr\_rcv\_topics
  - lbm\_context\_stats\_t\_stct, 129
- tr\_rcv\_unresolved\_topics
  - lbm\_context\_stats\_t\_stct, 129
- tr\_src\_topics
  - lbm\_context\_stats\_t\_stct, 129

- lbm\_context\_stats\_t\_stct, 129
- transport\_id
  - lbm\_transport\_source\_info\_t\_stct, 251
- transport\_idx
  - lbm\_transport\_source\_info\_t\_stct, 251
- tv\_secs
  - lbmpdm\_timestamp\_t, 304
- tv\_usecs
  - lbmpdm\_timestamp\_t, 304
- txw\_bytes
  - lbm\_src\_transport\_stats\_lbtrm\_t\_stct, 238
- txw\_msgs
  - lbm\_src\_transport\_stats\_lbtrm\_t\_stct, 238
- type
  - lbm\_apphdr\_chain\_elem\_t\_stct, 114
  - lbm\_async\_operation\_info\_t, 117
  - lbm\_msg\_t\_stct, 157
  - lbm\_rcv\_transport\_stats\_t\_stct, 197
  - lbm\_src\_event\_flight\_size\_notification\_t\_stct, 208
  - lbm\_src\_transport\_stats\_t\_stct, 245
  - lbm\_transport\_source\_info\_t\_stct, 252
  - lbm\_wildcard\_rcv\_stats\_t\_stct, 284
- u32
  - lbm\_hf\_sequence\_number\_t\_stct, 147
- u64
  - lbm\_hf\_sequence\_number\_t\_stct, 147
- uim\_dup\_msgs\_rcved
  - lbm\_context\_stats\_t\_stct, 129
- uim\_msgs\_no\_stream\_rcved
  - lbm\_context\_stats\_t\_stct, 130
- UME\_BLOCK\_DEBUG\_PRINT
  - umeblocksrc.h, 703
- UME\_BLOCK\_PRINT\_ERROR
  - umeblocksrc.h, 703
- ume\_block\_src\_create
  - umeblocksrc.h, 705
- ume\_block\_src\_delete
  - umeblocksrc.h, 705
- ume\_block\_src\_send\_ex
  - umeblocksrc.h, 705
- ume\_block\_src\_t\_stct, 306
- ume\_liveness\_receiving\_context\_t
  - lbm.h, 434
- ume\_liveness\_receiving\_context\_t\_stct, 307
- UME\_MALLOC\_RETURN
  - umeblocksrc.h, 703
- ume\_msg\_clientd
  - lbm\_src\_send\_ex\_info\_t\_stct, 232
- UME\_SEM\_DESTROY
  - umeblocksrc.h, 703
- UME\_SEM\_INIT
  - umeblocksrc.h, 703
- UME\_SEM\_POST
  - umeblocksrc.h, 704
- UME\_SEM\_TIMEDWAIT
  - umeblocksrc.h, 704
- UME\_SEM\_WAIT
  - umeblocksrc.h, 704
- umeblocksrc.h, 701
  - SLEEP\_MSEC, 702
  - UME\_BLOCK\_DEBUG\_PRINT, 703
  - UME\_BLOCK\_PRINT\_ERROR, 703
  - ume\_block\_src\_create, 705
  - ume\_block\_src\_delete, 705
  - ume\_block\_src\_send\_ex, 705
  - UME\_MALLOC\_RETURN, 703
  - UME\_SEM\_DESTROY, 703
  - UME\_SEM\_INIT, 703
  - UME\_SEM\_POST, 704
  - UME\_SEM\_TIMEDWAIT, 704
  - UME\_SEM\_WAIT, 704
- umq\_index
  - lbm\_src\_send\_ex\_info\_t\_stct, 232
- umq\_msg\_total\_lifetime
  - lbm\_umq\_msg\_total\_lifetime\_info\_t\_stct, 271
- umq\_total\_lifetime
  - lbm\_src\_send\_ex\_info\_t\_stct, 232
- unblock\_events
  - lbm\_event\_queue\_stats\_t\_stct, 144

---

unblock\_events\_tot  
    lbm\_event\_queue\_stats\_t\_stct, [144](#)

unrecovered\_tmo  
    lbm\_rcv\_transport\_stats\_lbtrm\_t\_stct, [186](#)  
    lbm\_rcv\_transport\_stats\_lbtru\_t\_stct, [193](#)

unrecovered\_twx  
    lbm\_rcv\_transport\_stats\_lbtrm\_t\_stct, [186](#)  
    lbm\_rcv\_transport\_stats\_lbtru\_t\_stct, [193](#)

username  
    lbm\_umm\_info\_t\_stct, [269](#)

value  
    lbm\_umq\_ulb\_application\_set\_attr\_t\_stct, [278](#)  
    lbm\_umq\_ulb\_receiver\_type\_attr\_t\_stct, [279](#)  
    lbmpdm\_field\_value\_stct\_t, [303](#)

value\_arr  
    lbmpdm\_field\_value\_stct\_t, [303](#)

wrcv\_msgs  
    lbm\_event\_queue\_stats\_t\_stct, [144](#)

wrcv\_msgs\_svc\_max  
    lbm\_event\_queue\_stats\_t\_stct, [144](#)

wrcv\_msgs\_svc\_mean  
    lbm\_event\_queue\_stats\_t\_stct, [144](#)

wrcv\_msgs\_svc\_min  
    lbm\_event\_queue\_stats\_t\_stct, [144](#)

wrcv\_msgs\_tot  
    lbm\_event\_queue\_stats\_t\_stct, [145](#)