



# **TIBCO Enterprise Message Service™ C and COBOL Reference**

*Software Release 8.5*

*May 2019*

*Document Updated: August 2020*

## Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, the TIBCO logo, the TIBCO O logo, Two-Second Advantage, TIBCO Cloud Integration, TIBCO Flogo Apps, TIBCO Flogo, TIB, Information Bus, TIBCO Enterprise Message Service, Rendezvous, and TIBCO Rendezvous are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Enterprise Java Beans (EJB), Java Platform Enterprise Edition (Java EE), Java 2 Platform Enterprise Edition (J2EE), and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle Corporation in the U.S. and other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

This software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. Please see the readme.txt file for the availability of this software version on a specific operating system platform.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

This and other products of TIBCO Software Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (<https://www.tibco.com/patents>) for details.

Copyright © 1997-2020. TIBCO Software Inc. All Rights Reserved.

# Contents

---

<b>About this Product</b> .....	<b>17</b>
<b>TIBCO Documentation and Support Services</b> .....	<b>18</b>
<b>Introduction</b> .....	<b>20</b>
Overview .....	20
Excluded Features and Restrictions .....	20
Naming Conventions and Length Limitations .....	20
Strings and Character Encodings .....	20
IBM z/OS and IBM i (i5)—COBOL and C .....	20
Configuring C and COBOL Clients for Fault-Tolerant Connections .....	20
<b>Messages</b> .....	<b>22</b>
Parts of a Message .....	22
Body Types .....	22
Headers .....	23
Properties .....	26
Set Message Properties .....	26
EMS Properties .....	26
JMS Properties .....	28
Message Selectors .....	28
Identifiers .....	28
Literals .....	28
Expressions .....	29
Operators .....	29
White Space .....	30
Data Type Conversion .....	30
tibemsMsg .....	31
tibemsMsg_Acknowledge .....	34
tibemsMsg_ClearBody .....	35
tibemsMsg_ClearProperties .....	35
tibemsMsg_Create .....	36
tibemsMsg_CreateCopy .....	37
tibemsMsg_CreateFromBytes .....	37
tibemsMsg_Destroy .....	38
tibemsMsg_GetAsBytes .....	39
tibemsMsg_GetAsBytesCopy .....	40
tibemsMsg_GetBodyType .....	41
tibemsMsg_GetByteSize .....	42

tibemsMsg_GetCorrelationID .....	43
tibemsMsg_GetDeliveryMode .....	44
tibemsMsg_GetDeliveryTime .....	44
tibemsMsg_GetDestination .....	45
tibemsMsg_GetEncoding .....	46
tibemsMsg_GetExpiration .....	47
tibemsMsg_GetMessageID .....	47
tibemsMsg_GetPriority .....	48
tibemsMsg_Get Property .....	49
tibemsMsg_GetPropertyNames .....	51
tibemsMsg_GetRedelivered .....	52
tibemsMsg_GetReplyTo .....	53
tibemsMsg_GetTimestamp .....	54
tibemsMsg_GetType .....	54
tibemsMsg_MakeWriteable .....	55
tibemsMsg_Print .....	56
tibemsMsg_PrintToBuffer .....	56
tibemsMsg_PropertyExists .....	57
tibemsMsg_Recover .....	58
tibemsMsg_SetCorrelationID .....	59
tibemsMsg_SetDeliveryMode .....	60
tibemsMsg_SetDestination .....	61
tibemsMsg_SetEncoding .....	61
tibemsMsg_SetExpiration .....	62
tibemsMsg_SetMessageID .....	63
tibemsMsg_SetPriority .....	64
tibemsMsg_Set Property .....	65
tibemsMsg_SetRedelivered .....	67
tibemsMsg_SetReplyTo .....	68
tibemsMsg_SetTimestamp .....	69
tibemsMsg_SetType .....	70
tibemsBytesMsg .....	71
tibemsBytesMsg_Create .....	71
tibemsBytesMsg_GetBodyLength .....	72
tibemsBytesMsg_GetBytes .....	73
tibemsBytesMsg_Read .....	73
tibemsBytesMsg_ReadBytes .....	76
tibemsBytesMsg_Reset .....	77
tibemsBytesMsg_SetBytes .....	77

tibemsBytesMsg_Write .....	78
tibemsBytesMsg_WriteBytes .....	80
tibemsMapMsg .....	81
tibemsMapMsg_Create .....	81
tibemsMapMsg_Get .....	82
tibemsMapMsg_GetMapNames .....	85
tibemsMapMsg_ItemExists .....	86
tibemsMapMsg_Set .....	86
tibemsMapMsg_SetBytes .....	90
tibemsObjectMsg .....	91
tibemsObjectMsg_Create .....	92
tibemsObjectMsg_GetObjectBytes .....	92
tibemsObjectMsg_SetObjectBytes .....	93
tibemsStreamMsg .....	94
tibemsStreamMsg_Create .....	95
tibemsStreamMsg_FreeField .....	95
tibemsStreamMsg_Read .....	96
tibemsStreamMsg_ReadBytes .....	98
tibemsStreamMsg_ReadField .....	99
tibemsStreamMsg_Reset .....	100
tibemsStreamMsg_Write .....	100
tibemsStreamMsg_WriteBytes .....	103
tibemsTextMsg .....	104
tibemsTextMsg_Create .....	104
tibemsTextMsg_GetText .....	105
tibemsTextMsg_SetText .....	106
tibemsData .....	106
tibemsDeliveryMode .....	107
tibemsNpCheckMode .....	108
tibemsMsgCompletionCallback .....	109
tibemsMsgEnum .....	110
tibemsMsgEnum_Destroy .....	110
tibemsMsgEnum_GetNextName .....	110
tibemsMsgField .....	111
tibemsMsgField_Print .....	113
tibemsMsgField_PrintToBuffer .....	114
tibemsMsgType .....	114
<b>Destination .....</b>	<b>116</b>
Destination Overview .....	116

tibemsDestinationType .....	117
tibemsDestination .....	118
tibemsDestination_Copy .....	118
tibemsDestination_Create .....	119
tibemsDestination_Destroy .....	120
tibemsDestination_GetName .....	120
tibemsDestination_GetType .....	121
tibemsQueue .....	122
tibemsQueue_Create .....	122
tibemsQueue_Destroy .....	123
tibemsQueue_GetQueueName .....	123
tibemsTemporaryQueue .....	124
tibemsTemporaryTopic .....	125
tibemsTopic .....	125
tibemsTopic_Create .....	126
tibemsTopic_Destroy .....	126
tibemsTopic_GetTopicName .....	127
<b>Consumer .....</b>	<b>128</b>
tibemsMsgConsumer .....	128
tibemsMsgConsumer_Close .....	128
tibemsMsgConsumer_GetDestination .....	129
tibemsMsgConsumer_GetMsgListener .....	130
tibemsMsgConsumer_GetMsgSelector .....	131
tibemsMsgConsumer_GetNoLocal .....	131
tibemsMsgConsumer_Receive .....	132
tibemsMsgConsumer_ReceiveNoWait .....	133
tibemsMsgConsumer_ReceiveTimeout .....	134
tibemsMsgConsumer_SetMsgListener .....	135
tibemsMsgCallback .....	135
<b>Producer .....</b>	<b>137</b>
tibemsMsgProducer .....	137
tibemsMsgProducer_AsyncSend .....	138
tibemsMsgProducer_Close .....	140
tibemsMsgProducer_GetDeliveryDelay .....	141
tibemsMsgProducer_GetDeliveryMode .....	142
tibemsMsgProducer_GetDestination .....	142
tibemsMsgProducer_GetDisableMessageID .....	143
tibemsMsgProducer_GetDisableMessageTimestamp .....	144
tibemsMsgProducer_GetNPSendCheckMode .....	144

tibemsMsgProducer_GetPriority .....	145
tibemsMsgProducer_GetTimeToLive .....	146
tibemsMsgProducer_Send .....	146
tibemsMsgProducer_SetDeliveryDelay .....	148
tibemsMsgProducer_SetDeliveryMode .....	149
tibemsMsgProducer_SetDisableMessageID .....	150
tibemsMsgProducer_SetDisableMessageTimestamp .....	151
tibemsMsgProducer_SetNPSendCheckMode .....	151
tibemsMsgProducer_SetPriority .....	152
tibemsMsgProducer_SetTimeToLive .....	153
<b>Requestor .....</b>	<b>155</b>
tibemsMsgRequestor .....	155
tibemsMsgRequestor_Close .....	155
tibemsMsgRequestor_Create .....	156
tibemsMsgRequestor_Request .....	157
<b>Connection .....</b>	<b>159</b>
tibemsConnection .....	159
tibemsConnection_Close .....	160
tibemsConnection_Create .....	161
tibemsConnection_CreateSession .....	162
tibemsConnection_GetActiveURL .....	163
tibemsConnection_GetClientId .....	164
tibemsConnection_GetExceptionListener .....	165
tibemsConnection_GetMetaData .....	165
tibemsConnection_IsDisconnected .....	166
tibemsConnection_SetClientId .....	167
tibemsConnection_SetExceptionListener .....	167
tibemsConnection_Start .....	168
tibemsConnection_Stop .....	169
tibemsConnectionMetaData .....	170
tibemsConnectionMetaData_GetEMS .....	171
tibemsConnectionMetaData_GetProvider .....	172
tibemsConnectionMetaData_GetEMSProviderName .....	173
tibemsExceptionCallback .....	174
tibemsMulticastExceptionCallback .....	174
tibemsSSL .....	175
tibemsSSL_GetTrace .....	176
tibemsSSL_OpenSSLVersion .....	176
tibemsSSL_SetTrace .....	177

tibemsSSLParams .....	178
tibemsSSLParams_AddIssuerCert .....	178
tibemsSSLParams_AddTrustedCert .....	179
tibemsSSLParams_Create .....	180
tibemsSSLParams_Destroy .....	180
tibemsSSLParams_GetIdentity .....	181
tibemsSSLParams_GetPrivateKey .....	182
tibemsSSLParams_SetAuthOnly .....	182
tibemsSSLParams_SetCiphers .....	183
tibemsSSLParams_SetExpectedHostName .....	183
tibemsSSLParams_SetHostNameVerifier .....	184
tibemsSSLParams_SetIdentity .....	185
tibemsSSLParams_SetPrivateKey .....	186
tibemsSSLParams_SetVerifyHost .....	186
tibemsSSLHostNameVerifier .....	187
<b>Connection Factory .....</b>	<b>189</b>
tibemsConnectionFactory .....	189
tibemsConnectionFactory_Create .....	191
tibemsConnectionFactory_CreateConnection .....	192
tibemsConnectionFactory_CreateXAConnection .....	193
tibemsConnectionFactory_Destroy .....	193
tibemsConnectionFactory_GetSSLProxyHost .....	194
tibemsConnectionFactory_GetSSLProxyPort .....	195
tibemsConnectionFactory_GetSSLProxyUser .....	195
tibemsConnectionFactory_GetSSLProxyPassword .....	196
tibemsConnectionFactory_Print .....	196
tibemsConnectionFactory_PrintToBuffer .....	197
tibemsConnectionFactory_SetClientID .....	198
tibemsConnectionFactory_SetConnectAttemptCount .....	198
tibemsConnectionFactory_SetConnectAttemptDelay .....	199
tibemsConnectionFactory_SetConnectAttemptTimeout .....	200
tibemsConnectionFactory_SetMetric .....	201
tibemsConnectionFactory_SetMulticastDaemon .....	201
tibemsConnectionFactory_SetMulticastEnabled .....	202
tibemsConnectionFactory_SetPkPassword .....	203
tibemsConnectionFactory_SetReconnectAttemptCount .....	203
tibemsConnectionFactory_SetReconnectAttemptDelay .....	204
tibemsConnectionFactory_SetReconnectAttemptTimeout .....	205
tibemsConnectionFactory_SetServerURL .....	206

tibemsConnectionFactory_SetSSLParams .....	206
tibemsConnectionFactory_SetSSLProxy .....	207
tibemsConnectionFactory_SetSSLProxyAuth .....	208
tibemsConnectionFactory_SetUserName .....	209
tibemsConnectionFactory_SetUserPassword .....	210
tibemsUFOConnectionFactory_Create .....	210
tibemsUFOConnectionFactory_CreateFromConnectionFactory .....	211
tibemsUFOConnectionFactory_RecoverConnection .....	211
tibemsFactoryLoadBalanceMetric .....	212
<b>Session .....</b>	<b>214</b>
tibemsSession .....	214
tibemsSession_Close .....	216
tibemsSession_Commit .....	217
tibemsSession_CreateBrowser .....	217
tibemsSession_CreateBytesMessage .....	218
tibemsSession_CreateConsumer .....	219
tibemsSession_CreateDurableSubscriber .....	220
tibemsSession_CreateMapMessage .....	222
tibemsSession_CreateMessage .....	222
tibemsSession_CreateProducer .....	223
tibemsSession_CreateSharedConsumer .....	224
tibemsSession_CreateSharedDurableConsumer .....	225
tibemsSession_CreateStreamMessage .....	227
tibemsSession_CreateTemporaryQueue .....	227
tibemsSession_CreateTemporaryTopic .....	228
tibemsSession_CreateTextMessage .....	229
tibemsSession_DeleteTemporaryQueue .....	230
tibemsSession_DeleteTemporaryTopic .....	231
tibemsSession_GetAcknowledgeMode .....	231
tibemsSession_GetTransacted .....	232
tibemsSession_Recover .....	233
tibemsSession_Rollback .....	234
tibemsSession_Unsubscribe .....	234
tibemsAcknowledgeMode .....	235
<b>Queue Browser .....</b>	<b>238</b>
tibemsQueueBrowser .....	238
tibemsQueueBrowser_Close .....	238
tibemsQueueBrowser_GetMsgSelector .....	239
tibemsQueueBrowser_GetNext .....	239

tibemsQueueBrowser_GetQueue .....	240
<b>Name Server Lookup .....</b>	<b>241</b>
tibemsLookupContext .....	241
tibemsLookupContext_Create .....	241
tibemsLookupContext_CreateExternal .....	243
tibemsLookupContext_Destroy .....	243
tibemsLookupContext_Lookup .....	244
tibemsLookupParams .....	245
tibemsLookupParams_Create .....	246
tibemsLookupParams_Destroy .....	246
tibemsLookupParams_GetLdapServerUrl .....	247
tibemsLookupParams_SetLdapBaseDN .....	247
tibemsLookupParams_SetLdapCAFile .....	248
tibemsLookupParams_SetLdapCAPath .....	248
tibemsLookupParams_SetLdapCertFile .....	249
tibemsLookupParams_SetLdapCiphers .....	249
tibemsLookupParams_SetLdapConnType .....	250
tibemsLookupParams_SetLdapCredential .....	250
tibemsLookupParams_SetLdapKeyFile .....	251
tibemsLookupParams_SetLdapPrincipal .....	251
tibemsLookupParams_SetLdapRandFile .....	252
tibemsLookupParams_SetLdapSearchScope .....	253
tibemsLookupParams_SetLdapServerUrl .....	253
<b>XA—External Transaction Manager .....</b>	<b>254</b>
tibemsXAConnection .....	254
tibemsXAConnection_Close .....	254
tibemsXAConnection_Create .....	255
tibemsXAConnection_CreateXASession .....	256
tibemsXAConnection_Get .....	257
tibemsXAConnection_GetXASession .....	258
XID .....	258
tibemsXAResource .....	258
tibemsXAResource_Commit .....	259
tibemsXAResource_End .....	260
tibemsXAResource_Forget .....	260
tibemsXAResource_GetRMID .....	261
tibemsXAResource_GetTransactionTimeout .....	262
tibemsXAResource_isSameRM .....	262
tibemsXAResource_Prepare .....	263

tibemsXAResource_Recover .....	264
tibemsXAResource_Rollback .....	265
tibemsXAResource_SetRMID .....	265
tibemsXAResource_SetTransactionTimeout .....	266
tibemsXAResource_Start .....	266
tibemsXASession .....	267
tibemsXASession_Close .....	268
tibemsXASession_GetSession .....	268
tibemsXASession_GetXAResource .....	269
<b>Types .....</b>	<b>270</b>
Uniform Types .....	270
<b>Utilities .....</b>	<b>271</b>
Utility Functions .....	271
tibems_Close .....	272
tibems_GetAllowCloseInCallback .....	273
tibems_GetConnectAttemptCount .....	274
tibems_GetConnectAttemptDelay .....	274
tibems_GetConnectAttemptTimeout .....	275
tibems_GetExceptionOnFTEvents .....	275
tibems_GetExceptionOnFTSwitch .....	276
tibems_GetMulticastDaemon .....	276
tibems_GetMulticastEnabled .....	277
tibems_GetReconnectAttemptCount .....	277
tibems_GetReconnectAttemptDelay .....	278
tibems_GetReconnectAttemptTimeout .....	278
tibems_GetSocketReceiveBufferSize .....	279
tibems_GetSocketSendBufferSize .....	279
tibems_IsConsumerMulticast .....	279
tibems_Open .....	280
tibems_SetAllowCloseInCallback .....	280
tibems_SetConnectAttemptCount .....	281
tibems_SetConnectAttemptDelay .....	282
tibems_SetConnectAttemptTimeout .....	283
tibems_SetExceptionOnFTEvents .....	283
tibems_setExceptionOnFTSwitch .....	284
tibems_SetMulticastDaemon .....	285
tibems_SetMulticastEnabled .....	286
tibems_SetMulticastExceptionListener .....	286
tibems_SetReconnectAttemptCount .....	287

tibems_SetReconnectAttemptDelay .....	288
tibems_SetReconnectAttemptTimeout .....	288
tibems_SetSocketReceiveBufferSize .....	289
tibems_SetSocketSendBufferSize .....	290
tibems_SetTraceFile .....	290
tibems_Sleep .....	291
tibems_Version .....	292
<b>Administration .....</b>	<b>293</b>
Administration Overview .....	293
tibemsAdmin .....	293
tibemsAdmin_Close .....	294
tibemsAdmin_Create .....	295
tibemsAdmin_GetCommandTimeout .....	296
tibemsAdmin_GetConsumer .....	296
tibemsAdmin_GetConsumers .....	297
tibemsAdmin_GetInfo .....	300
tibemsAdmin_GetProducerStatistics .....	301
tibemsAdmin_GetQueue .....	302
tibemsAdmin_GetQueues .....	303
tibemsAdmin_GetSubscriptions .....	304
tibemsAdmin_GetTopic .....	306
tibemsAdmin_GetTopics .....	307
tibemsAdmin_SetCommandTimeout .....	308
tibemsAdmin_SetExceptionListener .....	309
tibemsCollection .....	310
tibemsCollection_Destroy .....	310
tibemsCollection_GetCount .....	311
tibemsCollection_GetFirst .....	312
tibemsCollection_GetNext .....	312
tibemsConsumerInfo .....	313
tibemsConsumerInfo_Destroy .....	315
tibemsConsumerInfo_GetCreateTime .....	315
tibemsConsumerInfo_GetCurrentMsgCountSentByServer .....	316
tibemsConsumerInfo_GetCurrentMsgSizeSentByServer .....	317
tibemsConsumerInfo_GetDestinationName .....	317
tibemsConsumerInfo_GetDestinationType .....	318
tibemsConsumerInfo_GetDetailedStatistics .....	318
tibemsConsumerInfo_GetDurableName .....	319
tibemsConsumerInfo_GetElapsedSinceLastAcknowledged .....	320

tibemsConsumerInfo_GetElapsedSinceLastSent .....	321
tibemsConsumerInfo_GetID .....	321
tibemsConsumerInfo_GetPendingMessageCount .....	322
tibemsConsumerInfo_GetPendingMessageSize .....	322
tibemsConsumerInfo_GetSharedSubscriptionName .....	323
tibemsConsumerInfo_GetStatistics .....	324
tibemsConsumerInfo_GetTotalAcknowledgedCount .....	325
tibemsConsumerInfo_GetTotalMsgCountSentByServer .....	325
tibemsConsumerInfo_IsActive .....	326
tibemsConsumerInfo_IsConnected .....	327
tibemsConsumerInfo_IsConnectionConsumer .....	327
tibemsConsumerInfo_IsShared .....	328
tibemsDetailedDestStat .....	329
tibemsDetailedDestStat_GetDestinationName .....	329
tibemsDetailedDestStat_GetDestinationType .....	330
tibemsDetailedDestStat_GetStatData .....	330
tibemsProducerInfo .....	331
tibemsProducerInfo_Destroy .....	332
tibemsProducerInfo_GetCreateTime .....	332
tibemsProducerInfo_GetDestinationName .....	333
tibemsProducerInfo_GetDestinationType .....	333
tibemsProducerInfo_GetDetailedStatistics .....	334
tibemsProducerInfo_GetID .....	334
tibemsProducerInfo_GetStatistics .....	335
tibemsQueueInfo .....	336
tibemsQueueInfo_Create .....	337
tibemsQueueInfo_Destroy .....	338
tibemsQueueInfo_GetDeliveredMessageCount .....	338
tibemsQueueInfo_GetFlowControlMaxBytes .....	339
tibemsQueueInfo_GetInboundStatistics .....	339
tibemsQueueInfo_GetMaxBytes .....	340
tibemsQueueInfo_GetMaxMsgs .....	340
tibemsQueueInfo_GetName .....	341
tibemsQueueInfo_GetOutboundStatistics .....	342
tibemsQueueInfo_GetOverflowPolicy .....	342
tibemsQueueInfo_GetPendingMessageCount .....	343
tibemsQueueInfo_GetPendingMessageSize .....	344
tibemsQueueInfo_GetPendingPersistentMessageCount .....	344
tibemsQueueInfo_GetPendingPersistentMessageSize .....	345

tibemsQueueInfo_GetReceiverCount .....	345
tibemsServerInfo .....	346
tibemsServerInfo_Destroy .....	346
tibemsServerInfo_GetConsumerCount .....	347
tibemsServerInfo_GetProducerCount .....	347
tibemsServerInfo_GetQueueCount .....	348
tibemsServerInfo_GetTopicCount .....	348
tibemsStatData .....	349
tibemsStatData_GetByteRate .....	349
tibemsStatData_GetMessageRate .....	350
tibemsStatData_GetTotalBytes .....	351
tibemsStatData_GetTotalMessages .....	351
tibemsSubscriptionInfo .....	352
tibemsSubscriptionInfo_Destroy .....	353
tibemsSubscriptionInfo_GetConsumerCount .....	353
tibemsSubscriptionInfo_GetCreateTime .....	354
tibemsSubscriptionInfo_GetID .....	354
tibemsSubscriptionInfo_GetName .....	355
tibemsSubscriptionInfo_GetPendingMessageCount .....	356
tibemsSubscriptionInfo_GetPendingMessageSize .....	356
tibemsSubscriptionInfo_GetSelector .....	357
tibemsSubscriptionInfo_GetTopicName .....	357
tibemsSubscriptionInfo_HasSelector .....	358
tibemsSubscriptionInfo_IsDurable .....	359
tibemsSubscriptionInfo_IsShared .....	359
tibemsTopicInfo .....	360
tibemsTopicInfo_Create .....	361
tibemsTopicInfo_Destroy .....	362
tibemsTopicInfo_GetActiveDurableCount .....	362
tibemsTopicInfo_GetDurableCount .....	363
tibemsTopicInfo_GetDurableSubscriptionCount .....	363
tibemsTopicInfo_GetFlowControlMaxBytes .....	364
tibemsTopicInfo_GetInboundStatistics .....	365
tibemsTopicInfo_GetMaxBytes .....	365
tibemsTopicInfo_GetMaxMsgs .....	366
tibemsTopicInfo_GetName .....	367
tibemsTopicInfo_GetOutboundStatistics .....	367
tibemsTopicInfo_GetOverflowPolicy .....	368
tibemsTopicInfo_GetPendingMessageCount .....	369

tibemsTopicInfo_GetPendingMessageSize .....	369
tibemsTopicInfo_GetPendingPersistentMessageCount .....	370
tibemsTopicInfo_GetPendingPersistentMessageSize .....	370
tibemsTopicInfo_GetSubscriberCount .....	371
tibemsTopicInfo_GetSubscriptionCount .....	372
<b>Exception .....</b>	<b>373</b>
tibems_status .....	373
tibemsStatus_GetText .....	377
tibemsErrorContext .....	378
tibemsErrorContext_Create .....	379
tibemsErrorContext_Close .....	379
tibemsErrorContext_GetLastErrorString .....	380
tibemsErrorContext_GetLastErrorStackTrace .....	381
<b>IBM z/OS and IBM i .....</b>	<b>382</b>
IBM EBCDIC Platform Calls .....	382
tibems_SetCodePages .....	382
IBM z/OS Functions .....	383
tibx_MVSCConsole_SetConsumer .....	383
tibx_MVSCConsole_Create .....	384
Console_Response .....	385
SSL Implementation on IBM EBCDIC Systems .....	386
IBM System SSL Environment Variables on z/OS .....	386
Call Summary .....	386
tibemsSSL_System_GetTrace .....	387
tibemsSSL_System_SetFipsMode .....	388
tibemsSSL_System_SetTrace .....	389
tibemsSSL_System_Version .....	389
tibemsSSLParams_System_Create .....	390
tibemsSSLParams_System_Destroy .....	391
tibemsSSLParams_System_SetApplicationId .....	391
tibemsSSLParams_System_SetAuthOnly .....	392
tibemsSSLParams_System_SetCiphers .....	393
tibemsSSLParams_System_SetEnableTLS1 .....	397
tibemsSSLParams_System_SetEnableTLS11 .....	398
tibemsSSLParams_System_SetEnableTLS12 .....	399
tibemsSSLParams_System_SetExpectedHostName .....	400
tibemsSSLParams_System_SetKeyRingFile .....	400
tibemsSSLParams_System_SetLabel .....	401
tibemsSSLParams_System_SetLdapServerPassword .....	402

tibemsSSLParams_System_SetLdapServerUrl .....	403
tibemsSSLParams_System_SetLdapServerUserid .....	404
tibemsSSLParams_System_SetVerifyHostName .....	405

# About this Product

---

TIBCO is proud to announce the latest release of TIBCO Enterprise Message Service™ software.

This release is the latest in a long history of TIBCO products that leverage the power of the Information Bus® technology to enable truly event-driven IT environments. To find out more about how TIBCO Enterprise Message Service software and other TIBCO products are powered by TIB® technology, please visit us at [www.tibco.com](http://www.tibco.com).

TIBCO Enterprise Message Service software lets application programs send and receive messages according to the Java Message Service (JMS) protocol. It also integrates with TIBCO FTL, TIBCO Rendezvous, and TIBCO SmartSockets messaging products.

TIBCO EMS software is part of TIBCO® Messaging.

## Product Editions

TIBCO Messaging is available in a community edition and an enterprise edition.

TIBCO Messaging - Community Edition is ideal for getting started with TIBCO Messaging, for implementing application projects (including proof of concept efforts), for testing, and for deploying applications in a production environment. Although the community license limits the number of production clients, you can easily upgrade to the enterprise edition as your use of TIBCO Messaging expands.

The community edition is available free of charge. It is a full installation of the TIBCO Messaging software, with the following limitations and exclusions:

- Users may run up to 100 application instances or 1000 web/mobile instances in a production environment.
- Users do not have access to TIBCO Support, but you can use TIBCO Community as a resource (<https://community.tibco.com>).
- Available on Red Hat Enterprise Linux Server, Microsoft Windows & Windows Server and Apple macOS.

TIBCO Messaging - Community Edition has the following additional limitations and exclusions:

- Excludes Fault Tolerance of the server.
- Excludes Unshared State Failover.
- Excludes Routing of messages between servers.
- Excludes Central Administration.
- Excludes JSON configuration files.

TIBCO Messaging - Enterprise Edition is ideal for all application development projects, and for deploying and managing applications in an enterprise production environment. It includes all features presented in this documentation set, as well as access to TIBCO Support.

# TIBCO Documentation and Support Services

---

## How to Access TIBCO Documentation

Documentation for TIBCO products is available on the TIBCO Product Documentation website, mainly in HTML and PDF formats.

The TIBCO Product Documentation website is updated frequently and is more current than any other documentation included with the product. To access the latest documentation, visit <https://docs.tibco.com>.

## TIBCO Enterprise Message Service Documentation

The following documents for this product can be found on the [TIBCO Enterprise Message Service™](#) product documentation page:

- *TIBCO Enterprise Message Service User's Guide* Read this manual to gain an overall understanding of the product, its features, and configuration.
- *TIBCO Enterprise Message Service Central Administration* Read this manual for information on the central administration interface.
- *TIBCO Enterprise Message Service Installation* Read the relevant sections of this manual before installing this product.
- *TIBCO Enterprise Message Service C & COBOL Reference* The C API reference is available in HTML and PDF formats.
- *TIBCO Enterprise Message Service Java API Reference* The Java API reference can be accessed only through the HTML documentation interface.
- *TIBCO Enterprise Message Service .NET API Reference* The .NET API reference can be accessed only through the HTML documentation interface.
- *TIBCO Enterprise Message Service Installation on Red Hat OpenShift Container Platform* This manual describes how to run TIBCO Enterprise Message Service servers on the Red Hat® OpenShift Container Platform.
- *TIBCO Enterprise Message Service Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release. This document is available only in PDF format.

## Other TIBCO Product Documentation

You may find it useful to read the documentation for the following TIBCO products:

- TIBCO FTL®
- TIBCO Rendezvous®
- TIBCO SmartSockets®
- TIBCO EMS® Client for z/OS (CICS)
- TIBCO EMS® Client for z/OS (MVS)
- TIBCO EMS® Client for IBM i

## Third-Party Documentation

- Java™ Message Service specification, available through <http://www.oracle.com/technetwork/java/jms/index.html>.

- *Java™ Message Service* by Richard Monson-Haefel and David A. Chappell, O'Reilly and Associates, Sebastopol, California, 2001.
- Java™ Authentication and Authorization Service (JAAS) LoginModule Developer's Guide and Reference Guide, available through <http://www.oracle.com/technetwork/java/javase/jaas/index.html>.

### **How to Contact TIBCO Support**

You can contact TIBCO Support in the following ways:

- For an overview of TIBCO Support, visit <http://www.tibco.com/services/support>.
- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support portal at <https://support.tibco.com>.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to <https://support.tibco.com>. If you do not have a user name, you can request one by clicking Register on the website.

### **How to Join TIBCO Community**

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the [TIBCO Ideas Portal](#). For a free registration, go to <https://community.tibco.com>.

# Introduction

The following topics present concepts specific to the TIBCO Enterprise Message Service™ C API and COBOL API. For more information, see [TIBCO Enterprise Message Service User's Guide](#).

## Overview

TIBCO Enterprise Message Service Java API implements (and extends) the JMS 2.0 specification. TIBCO Enterprise Message Service C API closely mimics the Java API.

EMS COBOL API brings the power of EMS to z/OS systems. Its entry points parallel those of the C API, with identical function names and similar parameters.

## Excluded Features and Restrictions

This section summarizes features that are not available in either the C library, or the COBOL library.

### Feature Support

Feature	C	COBOL
XA protocols for external transaction managers	Yes	--
ConnectionConsumer, ServerSession, ServerSessionPool	--	--

## Naming Conventions and Length Limitations

The rules that govern naming conventions and length limitations in TIBCO Enterprise Message Service are described in the [TIBCO Enterprise Message Service User's Guide](#).

## Strings and Character Encodings

Typically, C clients manipulate strings using the character encoding of the machine on which they are running.

The EMS C client library itself does not do any encoding or decoding of characters. When sending a message, an EMS C client application can use `tibemsMsg_SetEncoding` to put information into the message describing the encoding used. When receiving a message in an EMS C client application, the encoding can be retrieved using `tibemsMsg_GetEncoding` and then a third party library can be used to do the actual decoding.

### Character Limits for Connection URLs

Connection URLs are limited to a length of 1000 characters. Note that this limit pertains to the C client library, as well as the administration tool and the `factories.conf` configuration file.

## IBM z/OS and IBM i (i5)—COBOL and C

In EBCDIC environments, the EMS client library automatically converts message strings. To client programs, all strings appear in the host code page. On the network, all strings appear in the network host page. For details, see `tibems_SetCodePages()`

All the COBOL calls documented in this reference refer to COBOL on the z/OS and IBM i platforms.

## Configuring C and COBOL Clients for Fault-Tolerant Connections

When connecting a fault-tolerant client to EMS, you must specify two or more EMS servers.

C clients list the primary and backup server URLs in the `brokerURL` argument to a connection constructor, separated by a comma.

**Example**

In this example, the first server is `tcp://server0:7222`, and the second server is `tcp://server1:7344` (if first server is not available).

```
tibemsConnection_Create(  
    &connection,  
    "tcp://server0:7222,  
    tcp://server1:7344",  
    NULL, "admin", NULL );
```

See:

*Configure Clients for Shared State Failover Connections in the TIBCO Enterprise Message Service User's Guide.*

# Messages

---

Message objects carry application data between client program processes. The following topics present the structure of JMS message selector syntax to specify a subset of messages based on their property values, the message types, and their functions.

## Parts of a Message

Messages consist of three parts:

- **Body**

The body of a message bears the information content of an application. Several types of message body organize that information in different ways; see [Body Types](#).

- **Header**

Headers associate a fixed set of header field names with values. Clients and providers use headers to identify and route messages.

- **Properties**

Properties associate an extensible set of property names with values. The EMS server uses properties to attach ancillary information to messages. Client applications can also use properties—for example, to customize message filtering.

## Body Types

EMS follows JMS in defining five types of message body:

- [tibemsMapMsg](#)

The message body is a mapping from field names to values. Field names are strings. EMS supports an extended set of values types (extending JMS). Programs can access fields either by name, or sequentially (though the order of that sequence is indeterminate).

- [tibemsObjectMsg](#)

The message body is one serializable object.

- [tibemsStreamMsg](#)

The message body is a stream of values. Programs write the values sequentially into the stream, and read values sequentially from the stream.

- [tibemsTextMsg](#)

The message body is one character string (of any length). This text string can represent any text, including an XML document.

- [tibemsBytesMsg](#)

The message body is a stream of uninterpreted bytes. Programs can use this body type to emulate body types that do not map naturally to one of the other body types.

### See Also

[tibemsMsgType](#)

[tibemsMsg\\_GetBodyType](#)

## Headers

JMS Message headers associate a fixed set of header field names with values. Clients and providers use headers to identify and route messages.

Programs can access header values using the function calls in the following table.

However, programs can effectively set only three message header properties—Reply To, Correlation ID and Type. For all other header properties, the provider ignores or overwrites values set by client programs.

Description
<p><b>Correlation ID</b></p> <p>Correlation ID refers to a related message. For example, when a consumer responds to a request message by sending a reply, it can set the correlation ID of the reply to indicate the request message.</p> <p>The JMS specification allows three categories of values for the correlation ID property:</p> <ul style="list-style-type: none"> <li>• <b>Message ID</b> <p>A message ID is a unique string that the provider assigns to a message. Programs can use these IDs to correlate messages. For example, a program can link a response to a request by setting the correlation ID of a response message to the message ID of the corresponding request message.</p> <p>Message ID strings begin with the prefix ID: (which is reserved for this purpose).</p> </li> <li>• <b>String</b> <p>Programs can also correlate messages using arbitrary strings, with semantics determined by the application.</p> <p>These strings must <i>not</i> begin with the prefix ID: (which is reserved for message IDs).</p> </li> <li>• <b>Byte Array</b> <p>This implementation does not support byte array values for the correlation ID property. The JMS specification does not require support.</p> </li> </ul> <p><code>tibemsMsg_GetCorrelationID</code></p> <p><code>tibemsMsg_SetCorrelationID</code></p>
<p><b>Delivery Delay</b></p> <p>A producer can specify that a message must not be delivered until after a specified time interval, which directs the server to delay delivery of the message.</p> <p>For detailed information on delivery delay, see the <i>TIBCO Enterprise Message Service User's Guide</i>.</p> <p>Sending calls can set the delivery delay for each message based on a property of the producer. To set the producer property, see <code>tibemsMsgProducer_SetDeliveryDelay</code>.</p> <p><code>tibemsMsg_GetDeliveryTime</code></p>

Description
<p><b>Delivery Mode</b></p> <p>Delivery Mode instructs the server concerning persistent storage for the message. For detailed information on delivery modes, see the <a href="#">TIBCO Enterprise Message Service User's Guide</a>.</p> <p>Sending calls set the delivery mode for each message, based on either a property of the producer, or on a parameter to the sending call. To set the producer property, see <a href="#">tibemsMsgProducer_SetDeliveryMode</a>.</p> <p><a href="#">tibemsMsg_GetDeliveryMode</a></p> <p><a href="#">tibemsMsg_SetDeliveryMode</a></p> <p>For values, see <a href="#">tibemsDeliveryMode</a>.</p>
<p><b>Destination</b></p> <p>Sending calls set the destination (queue or topic) of the message in this header and will overwrite any existing value. The value is based on either a property of the producer, or on a parameter to the send call.</p> <p>Listeners that consume messages from wildcard destinations can use this property to determine the actual destination of a message.</p> <p><a href="#">tibemsMsg_GetDestination</a></p> <p><a href="#">tibemsMsg_SetDestination</a></p>
<p><b>Expiration</b></p> <p>Sending calls set the expiration time (in milliseconds) of the message in this field:</p> <ul style="list-style-type: none"> <li>• If the time-to-live is non-zero, the expiration is the sum of that time-to-live and the sending client's current time (GMT).</li> <li>• If the time-to-live is zero, then expiration is also zero—indicating that the message never expires.</li> </ul> <p>The server discards a message when its expiration time has passed. However, the JMS specification does not guarantee that clients do not receive expired messages.</p> <p><a href="#">tibemsMsg_GetExpiration</a></p> <p><a href="#">tibemsMsg_SetExpiration</a></p>
<p><b>Message ID</b></p> <p>Sending calls assign a unique ID to each message, and record it in this header.</p> <p>All message ID values start with the 3-character prefix ID: (which is reserved for this purpose).</p> <p>Applications that do not require message IDs can reduce overhead costs by disabling IDs; see <a href="#">tibemsMsgProducer_SetDisableMessageID</a>. When the producer disables IDs, the value of this header is null.</p> <p><a href="#">tibemsMsg_GetMessageID</a></p> <p><a href="#">tibemsMsg_SetMessageID</a></p>

Description
<p><b>Priority</b></p> <p>Sending calls set the priority of a message in this header, based on either a property of the producer (<a href="#">tibemsMsgProducer_SetPriority</a>), or on a parameter to the send call.</p> <p>The JMS specification defines ten levels of priority value, from zero (lowest priority) to 9 (highest priority). The specification suggests that clients consider 0–4 as gradations of normal priority, and priorities 5–9 as gradations of expedited priority.</p> <p>Priority affects the order in which the server delivers messages to consumers (higher values first). The JMS specification does not require all providers to implement priority ordering of messages. (EMS supports priorities, but other JMS providers might not.)</p> <p><a href="#">tibemsMsg_GetPriority</a></p> <p><a href="#">tibemsMsg_SetPriority</a></p>
<p><b>Redelivered</b></p> <p>The server sets this header to indicate whether a message might duplicate a previously delivered message:</p> <ul style="list-style-type: none"> <li>• false—The server has <i>not</i> previously attempted to deliver this message to the consumer.</li> <li>• true—It is likely (but not guaranteed) that the server has previously attempted to deliver this message to the consumer, but the consumer did not return timely acknowledgment.</li> </ul> <p><a href="#">tibemsMsg_GetRedelivered</a></p> <p><a href="#">tibemsMsg_SetRedelivered</a></p> <p>See also, <a href="#">tibemsAcknowledgeMode</a></p>
<p><b>Reply To</b></p> <p>Sending clients can set this header to request that recipients reply to the message:</p> <ul style="list-style-type: none"> <li>• When the value is a destination object, recipients can send replies to that destination. Such a message is called a <i>request</i>.</li> <li>• When the value is null, the sender does not expect a reply.</li> </ul> <p>When sending a reply, clients can refer to the corresponding request by setting the Correlation ID.</p> <p><a href="#">tibemsMsg_GetReplyTo</a></p> <p><a href="#">tibemsMsg_SetReplyTo</a></p>
<p><b>Timestamp</b></p> <p>Sending calls record a UTC timestamp in this header, indicating the approximate time that the server accepted the message.</p> <p>The value is in milliseconds since January 1, 1970 (as in Java).</p> <p>Applications that do not require timestamps can reduce overhead costs by disabling timestamps; see <a href="#">tibemsMsgProducer_SetDisableMessageTimestamp</a>. When the producer disables timestamps, the value of this header is zero.</p> <p><a href="#">tibemsMsg_GetTimestamp</a></p> <p><a href="#">tibemsMsg_SetTimestamp</a></p>

Description
<p><b>Type</b></p> <p>Some JMS providers use a message repository to store message type definitions. Client programs can store a value in this field to reference a definition in the repository. EMS support this header, but does not use it.</p> <p>The JMS specification does not define a standard message definition repository, nor does it define a naming policy for message type definitions.</p> <p>Some providers require message type definitions for each application message. To ensure compatibility with such providers, client programs can set this header, even if the client application does not use it.</p> <p>To ensure portability, clients can set this header with symbolic values (rather than literals), and configure them to match the provider's repository.</p> <p><a href="#">tibemsMsg_GetType</a></p> <p><a href="#">tibemsMsg_SetType</a></p>

## Properties

Properties associate an extensible set of property field names with values. The EMS server uses properties to attach ancillary information to messages.

Client applications can also use properties—for example, to customize message filtering; see [Message Selectors](#).

## Set Message Properties

Property names must conform to the syntax for message selector identifiers.

For more information on the syntax, see [Identifiers](#).

Property values cannot be null or an empty string.

Sending programs can set property values before sending a message.

Receiving programs cannot ordinarily set property values on inbound messages. However, [tibemsMsg\\_ClearProperties](#) removes all existing the properties from a message, and lets the program set property values.

## EMS Properties

The JMS specification reserves the property name prefix `JMS_vendor_name_` for provider-specific properties (for EMS, this prefix is `JMS_TIBCO_`). Properties that begin with this prefix refer to features of EMS; client programs may use these properties to access those features, but not for communicating application-specific information among client programs.

Property	Description
JMS_TIBCO_CM_PUBLISHER	Correspondent name of an RVCN sender for messages imported from TIBCO Rendezvous.
JMS_TIBCO_CM_SEQUENCE	Sequence number of an RVCN message imported from TIBCO Rendezvous.

Property	Description
JMS_TIBCO_COMPRESS	Senders may set this property to request that EMS compress the message before sending it to the server. The .NET client API does not support this feature at this time.
JMS_TIBCO_DISABLE_SENDER	Senders may set this property to prevent the EMS server from including the sender name in the message when the server sends it to consumers; see <a href="#">JMS_TIBCO_SENDER</a> .
JMS_TIBCO_IMPORTED	When the EMS server imports a message from an external message service (such as TIBCO Rendezvous or TIBCO SmartSockets), it sets this property to true.
JMS_TIBCO_MSG_EXT	Producers can set this property to true to indicate that a message uses EMS extensions to the JMS specification for messages.  The server sets this property to true when importing a message from an external message service, since the message might use those extensions.
JMS_TIBCO_MSG_TRACE	When a producer sets this property, the EMS server generates trace output when the message arrives from the producer, and whenever a consumer receives it. <ul style="list-style-type: none"> <li>When the property value is null, the trace output contains the message ID and sequence number.</li> <li>When the property value is body, the trace output includes the message body as well.</li> </ul>
JMS_TIBCO_PRESERVE_UNDELIVERED	When this property is true, the server preserves a record of undeliverable messages by delivering them to the undelivered message queue, <code>\$sys.undelivered</code> .
JMS_TIBCO_SENDER	The EMS server fills this property with the <i>user name</i> (string) of the client that sent the message. This feature applies only when the <code>sender_name</code> property of the message's destination is non-null. The sender can disable this feature (overriding the destination property <code>sender_name</code> ) by setting a non-null value for the message property <a href="#">JMS_TIBCO_DISABLE_SENDER</a> .
JMS_TIBCO_SS_SENDER	When the EMS server imports a message from TIBCO SmartSockets, it sets this property to the SmartSockets sender header field (in SmartSockets syntax).

## COBOL Constants

```

01  TIBEMS-PROPERTIES.
    05  JMS-TIBCO-CM-PUBLISHER          PIC X(23) VALUE
        Z'JMS_TIBCO_CM_PUBLISHER' .
    05  JMS-TIBCO-CM-SEQUENCE          PIC X(22) VALUE
        Z'JMS_TIBCO_CM_SEQUENCE' .
    05  JMS-TIBCO-COMPRESS              PIC X(19) VALUE

```

```

Z 'JMS_TIBCO_COMPRESS' .
05 JMS-TIBCO-DISABLE-SENDER          PIC X(25) VALUE
Z 'JMS_TIBCO_DISABLE_SENDER' .
05 JMS-TIBCO-IMPORTED                PIC X(19) VALUE
Z 'JMS_TIBCO_IMPORTED' .
05 JMS-TIBCO-MSG-EXT                 PIC X(19) VALUE
Z 'JMS_TIBCO_MSG_EXT' .
05 JMS-TIBCO-MSG-TRACE               PIC X(20) VALUE
Z 'JMS_TIBCO_MSG_TRACE' .
05 JMS-TIBCO-PRESERVE-UNDELIVERED    PIC X(31) VALUE
Z 'JMS_TIBCO_PRESERVE_UNDELIVERED' .
05 JMS-TIBCO-SENDER                  PIC X(17) VALUE
Z 'JMS_TIBCO_SENDER' .
05 JMS-TIBCO-SS-SENDER                PIC X(20) VALUE
Z 'JMS_TIBCO_SS_SENDER' .

```

## JMS Properties

The JMS specification reserves the property name prefix `JMSX` for properties defined by JMS. Client programs may use these properties to access those features, but not for communicating application-specific information among client programs.

For information about these properties, see the JMS specification.

## Message Selectors

A message selector is string that lets a client program specify a set of messages, based on the values of message headers and properties. A selector *matches* a message if, after substituting header and property values from the message into the selector string, the string evaluates to true. Consumers can request that the server deliver only those messages that match a selector.

The syntax of selectors is based on a subset of SQL92 conditional expression syntax.

## Identifiers

Identifiers can refer to the values of message headers and properties, but not to the message body. Identifiers are case-sensitive.

- **Basic Syntax**

An identifier is a sequence of letters and digits, of any length, that begins with a letter. As in Java, the set of letters includes `_` (underscore) and `$` (dollar).

- **Illegal**

Certain names are exceptions, which cannot be used as identifiers. In particular, `NULL`, `TRUE`, `FALSE`, `NOT`, `AND`, `OR`, `BETWEEN`, `LIKE`, `IN`, `IS`, and `ESCAPE` are defined to have special meaning in message selector syntax.

- **Value**

Identifiers refer either to message header names or property names. The type of an identifier in a message selector corresponds to the type of the header or property value. If an identifier refers to a header or property that does not exist in a message, its value is `NULL`.

## Literals

- **String Literal**

A string literal is enclosed in single quotes. To represent a single quote within a literal, use two single quotes; for example, `'literal' 's'`. String literals use the Unicode character encoding. String literals are case sensitive.

- **Exact Numeric Literal**

An exact numeric literal is a numeric value without a decimal point, such as `57`, `-957`, and `+62`; numbers in the range of long are supported.

- **Approximate Numeric Literal**

An approximate numeric literal is a numeric value with a decimal point (such as `7.`, `-95.7`, and `+6.2`), or a numeric value in scientific notation (such as `7E3` and `-57.9E2`); numbers in the range of double are supported. Approximate literals use the floating-point literal syntax of the Java programming language.

- **Boolean Literal**

The boolean literals are `TRUE` and `FALSE` (case insensitive).

Internal computations of expression values use a 3-value boolean logic similar to SQL. However, the final value of an expression is always either `TRUE` or `FALSE`—never `UNKNOWN`.

## Expressions

- **Selectors as Expressions**

Every selector is a conditional expression. A selector that evaluates to true matches the message; a selector that evaluates to false or unknown does not match.

- **Arithmetic Expression**

Arithmetic expressions are composed of numeric literals, identifiers (that evaluate to numeric literals), arithmetic operations, and smaller arithmetic expressions.

- **Conditional Expression**

Conditional expressions are composed of comparison operations, logical operations, and smaller conditional expressions.

- **Order of Evaluation**

Order of evaluation is left-to-right, within precedence levels. Parentheses override this order.

## Operators

- **Case Insensitivity**

Operator names are case-insensitive.

- **Logical Operators**

Logical operators in precedence order: `NOT`, `AND`, `OR`.

- **Comparison Operators**

Comparison operators: `=`, `>`, `>=`, `<`, `<=`, `<>` (not equal).

These operators can compare only values of comparable types. (Exact numeric values and approximate numerical values are comparable types.) Attempting to compare incomparable types yields false. If either value in a comparison evaluates to `NULL`, then the result is unknown (in SQL 3-valued logic).

Comparison of string values is restricted to `=` and `<>`. Two strings are equal if and only if they contain the same sequence of characters. Comparison of boolean values is restricted to `=` and `<>`.

- **Arithmetic Operators**

Arithmetic operators in precedence order:

- `+`, `-` (unary)
- `*`, `/` (multiplication and division)
- `+`, `-` (addition and subtraction)

Arithmetic operations obey numeric promotion rules of the Java programming language.

- **Between Operator**

*arithmetic-expr1* [NOT] BETWEEN *arithmetic-expr2* AND *arithmetic-expr3*

The BETWEEN comparison operator includes its endpoints. For example:

- `age BETWEEN 5 AND 9` is equivalent to `age >= 5 AND age <= 9`
- `age NOT BETWEEN 5 AND 9` is equivalent to `age < 5 OR age > 9`

- **String Set Membership**

*identifier* [NOT] IN (*string-literal1*, *string-literal2*, ...)

The *identifier* must evaluate to either a string or NULL. If it is NULL, then the value of this expression is unknown. You can use a maximum of 32,767 string-literals in the string set.

- **Pattern Matching**

*identifier* [NOT] LIKE *pattern-value* [ESCAPE *escape-character*]

The *identifier* must evaluate to a string.

The *pattern-value* is a string literal, in which some characters bear special meaning:

- `_` (underscore) can match any single character.
- `%` (percent) can match any sequence of zero or more characters.
- *escape-character* preceding either of the special characters changes them into ordinary characters (which match only themselves).

- **Null Header or Property**

*identifier* IS NULL

This comparison operator tests whether a message header is null, or a message property is absent.

*identifier* IS NOT NULL

This comparison operator tests whether a message header or message property is non-null.

## White Space

White space is any of the characters space, horizontal tab, form feed, or line terminator—or any contiguous run of characters in this set.

## Data Type Conversion

The following table summarizes legal datatype conversions. The symbol X in indicates that a value written into a message as the row type can be extracted as the column type. This table applies to all message values—including map pairs, headers and properties—except as noted below.

	bool	byte	short	char	int	long	float	double	string	byte[]
bool	X								X	
byte		X	X		X	X			X	
short			X		X	X			X	
char				X					X	
int					X	X			X	

	bool	byte	short	char	int	long	float	double	string	byte[]
long						X			X	
float							X	X	X	
double								X	X	
string	X	X	X		X	X	X	X	X	
byte[]										X

### Notes

- Message properties cannot have byte array values.
- Values written as strings can be extracted as a numeric or boolean type only when it is possible to parse the string as a number of that type.

## tibemsMsg

*Type*

### Purpose

Messages that carry information among EMS client programs.

### Related Types

[tibemsBytesMsg](#), [tibemsMapMsg](#), [tibemsObjectMsg](#), [tibemsStreamMsg](#), [tibemsTextMsg](#)

Function	Description
<b>Receipt</b>	
<a href="#">tibemsMsg_Acknowledge</a>	Acknowledge messages.
<b>Body</b>	
<a href="#">tibemsMsg_GetBodyType</a>	Get the body type of a message.
<b>Message Life Cycle</b>	
<a href="#">tibemsMsg_ClearBody</a>	Clear the body of a message.
<a href="#">tibemsMsg_ClearProperties</a>	Clear the properties of a message.
<a href="#">tibemsMsg_Create</a>	Create a message object.
<a href="#">tibemsMsg_CreateCopy</a>	Create a copy of the message object.
<a href="#">tibemsMsg_CreateFromBytes</a>	Create a message object from data in a byte sequence.
<a href="#">tibemsMsg_Destroy</a>	Destroy a message.

Function	Description
<code>tibemsMsg_MakeWriteable</code>	Make a message writable.
<code>tibemsMsg_Recover</code>	Recover a single message.
<b>Output</b>	
<code>tibemsMsg_GetAsBytes</code>	Get a byte sequence representation of the message object.
<code>tibemsMsg_GetAsBytesCopy</code>	Copy a byte sequence representation of the message object into storage supplied by the program.
<code>tibemsMsg_GetByteSize</code>	Compute the size of the byte sequence representation of the message.
<code>tibemsMsg_Print</code>	Print a message.
<code>tibemsMsg_PrintToBuffer</code>	Print a message into a buffer.
<b>Headers and Properties</b>	
For details, see <a href="#">Headers</a> .	
<code>tibemsMsg_ClearProperties</code>	Clear the properties of a message.
<code>tibemsMsg_GetCorrelationID</code>	Get the correlation ID header of a message.
<code>tibemsMsg_GetDeliveryMode</code>	Get the delivery mode header from a message.
<code>tibemsMsg_GetDeliveryTime</code>	Get the delivery time header from a message.
<code>tibemsMsg_GetDestination</code>	Get the destination header from a message.
<code>tibemsMsg_GetEncoding</code>	Get the character encoding header from a message.
<code>tibemsMsg_GetExpiration</code>	Get the expiration header from a message.
<code>tibemsMsg_GetMessageID</code>	Get the message ID header from a message.
<code>tibemsMsg_GetPriority</code>	Get the priority header from a message.
<code>tibemsMsg_Get Property</code>	Get the value of a message property.
<code>tibemsMsg_GetPropertyNames</code>	Get a list of property names from a message.
<code>tibemsMsg_GetRedelivered</code>	Get the redelivered header from a message.
<code>tibemsMsg_GetReplyTo</code>	Get the reply-to header from a message.
<code>tibemsMsg_GetTimestamp</code>	Get the timestamp header from a message.

Function	Description
<a href="#">tibemsMsg_GetType</a>	Get the type header of a message.
<a href="#">tibemsMsg_PropertyExists</a>	Test whether a named property has been set on a message.
<a href="#">tibemsMsg_SetCorrelationID</a>	Set the correlation ID header of a message.
<a href="#">tibemsMsg_SetDeliveryMode</a>	Set the delivery mode header of a message.
<a href="#">tibemsMsg_SetDestination</a>	Set the destination header of a message.
<a href="#">tibemsMsg_SetEncoding</a>	Set the character encoding header of a message.
<a href="#">tibemsMsg_SetExpiration</a>	Set the expiration header of a message.
<a href="#">tibemsMsg_SetMessageID</a>	Set the message ID header of a message.
<a href="#">tibemsMsg_SetPriority</a>	Set the priority header of a message.
<a href="#">tibemsMsg_Set Property</a>	Set the value of a message property.
<a href="#">tibemsMsg_SetRedelivered</a>	Set the redelivered header of a message.
<a href="#">tibemsMsg_SetReplyTo</a>	Set the reply-to header of a message.
<a href="#">tibemsMsg_SetTimestamp</a>	Set the timestamp header of a message.
<a href="#">tibemsMsg_SetType</a>	Set the type header of a message.

### *Message Constants*

Constant	Description
TIBEMS_DEFAULT_DELIVERY_MODE	<a href="#">TIBEMS_PERSISTENT</a> When neither the sending call nor the producer supplies a delivery mode, this default applies.
TIBEMS_DEFAULT_PRIORITY	4 When neither the sending call nor the producer supplies a priority, this default applies. See also, <a href="#">Priority</a> .
TIBEMS_DEFAULT_TIME_TO_LIVE	0 When neither the sending call nor the producer supplies a priority, this default applies. The default value, zero, indicates that messages do not expire. See also <a href="#">Expiration</a> .

## tibemsMsg\_Acknowledge

Function

### Purpose

Acknowledge messages.

### C Declaration

```
tibems_status tibemsMsg_Acknowledge(
    tibemsMsg message );
```

### COBOL Call

```
CALL "tibemsMsg_Acknowledge"
    USING BY VALUE message,
    RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Acknowledge this message (but for the actual behavior of this call, see the Remarks below).

### Remarks

The behavior of this call depends on the acknowledgment mode of the [tibemsSession](#).

- In [TIBEMS\\_CLIENT\\_ACKNOWLEDGE](#) mode, this call acknowledges *all* messages that the program has consumed within the *session*. (This behavior complies with the JMS specification.)
- In [TIBEMS\\_EXPLICIT\\_CLIENT\\_ACKNOWLEDGE](#) mode, this call acknowledges *only* the individual message. (This mode and behavior are proprietary extensions, specific to TIBCO EMS.)
- In [TIBEMS\\_EXPLICIT\\_CLIENT\\_DUPS\\_OK\\_ACKNOWLEDGE](#) mode, this call lazily acknowledges *only* the individual message. *Lazy* means that the provider client library can delay transferring the acknowledgment to the server until a convenient time; meanwhile the server might redeliver the message. (This mode and behavior are proprietary extensions, specific to TIBCO EMS.)
- In all other modes, this call has no effect. In particular, modes that specify transactions or implicit acknowledgment do not require the consuming program to call this function. However, calling it does not produce an exception. (This behavior complies with the JMS specification.)

### Consumed

Two events mark a message as *consumed*—that is, eligible for acknowledgment using this function:

- Just before the provider calls an [tibemsMsgCallback](#) function, it marks the message argument as consumed.
- Just before a receive call returns a message, it marks that message as consumed.

### Redelivery

The server might redeliver unacknowledged messages.

**Restriction**

It is illegal to call this function after closing the session, the connection or the consumer through which the message arrived.

**See Also**

[tibemsMsgConsumer\\_Receive](#)

[tibemsSession](#)

[tibemsAcknowledgeMode](#)

**tibemsMsg\_ClearBody**

*Function*

**Purpose**

Clear the body of a message.

**C Declaration**

```
tibems_status tibemsMsg_ClearBody(
    tibemsMsg message );
```

**COBOL Call**

```
CALL "tibemsMsg_ClearBody"
    USING BY VALUE message,
    RETURNING tibems-status
END-CALL.
```



message has usage pointer.

**Parameters**

Parameter	Description
message	Message to be cleared.

**Remarks**

Clearing the body of a message leaves its header and property values unchanged.

If the message body was read-only, this function makes it writeable. The message body appears and behaves identically to an empty body in a newly created message.

**tibemsMsg\_ClearProperties**

*Function*

**Purpose**

Clear the properties of a message.

**C Declaration**

```
tibems_status tibemsMsg_ClearProperties(
    tibemsMsg message );
```

## COBOL Call

```
CALL "tibemsMsg_ClearProperties"
    USING BY VALUE message,
        RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Message to be cleared.

### Remarks

Clearing the property values of a message leaves its header values and body unchanged.

## tibemsMsg\_Create

*Function*

### Purpose

Create a message object.

### C Declaration

```
tibems_status tibemsMsg_Create(
    tibemsMsg* message );
```

## COBOL Call

```
CALL "tibemsMsg_Create"
    USING BY REFERENCE message,
        RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Store a pointer to the new message.

### Remarks

Create a new message.

When your application creates a message, it also allocates storage for that message. This storage must subsequently be freed by a call to `tibemsMsg_Destroy`.

### See Also

[tibemsMsg\\_Destroy](#)

## tibemsMsg\_CreateCopy

*Function*

### Purpose

Create a copy of the message object.

### C Declaration

```
tibems_status tibemsMsg_CreateCopy(
    const tibemsMsg message,
    tibemsMsg* copy );
```

### COBOL Call

```
CALL "tibemsMsg_CreateCopy"
    USING BY VALUE message,
         BY REFERENCE copy,
         RETURNING tibems-status
END-CALL.
```



message and copy have usage pointer.

### Parameters

Parameter	Description
message	Copy this message.
copy	Store a pointer to the new copy.

### Remarks

Create a new message by copying an existing message.

The copy is completely independent of the original message. Pointer data in fields are independent copies of the original values.

This function copies the entire message, including headers, properties, and body data.

This function allocates the storage for the copy. The duration of the copy is independent of the original message. Your program owns the messages that it creates, and must destroy those messages to reclaim the storage. That is, each call to this function must be paired with a call to [tibemsMsg\\_Destroy](#).

### See Also

[tibemsMsg\\_Destroy](#)

## tibemsMsg\_CreateFromBytes

*Function*

### Purpose

Create a message object from data in a byte sequence.

### C Declaration

```
tibems_status tibemsMsg_CreateFromBytes(
    tibemsMsg* message,
    const void* bytes );
```

## COBOL Call

```
CALL "tibemsMsg_CreateFromBytes"
    USING BY REFERENCE message,
          BY REFERENCE bytes,
          RETURNING tibems-status
END-CALL.
```



message has usage pointer.

## Parameters

Parameter	Description
message	Store a pointer to the new message.
bytes	Create a message from the data in this byte sequence. This data buffer represents the message in EMS wire format. To produce this type of buffer, use either <a href="#">tibemsMsg_GetAsBytes</a> or <a href="#">tibemsMsg_GetAsBytesCopy</a> .

## Remarks

Create a new message from a byte sequence and populates the message with data.

This function allocates the storage for the new message. Your program owns the messages that it creates, and must destroy those messages to reclaim the storage. That is, each call to this function must be paired with a call to [tibemsMsg\\_Destroy](#).

The new message is independent of the original byte sequence. They do not share any storage.

The newly created message is read-only; to enable modification without erasing the content, call [tibemsMsg\\_MakeWriteable](#).

## See Also

[tibemsMsg\\_Destroy](#)

[tibemsMsg\\_GetAsBytes](#)

[tibemsMsg\\_GetAsBytesCopy](#)

[tibemsMsg\\_MakeWriteable](#)

## tibemsMsg\_Destroy

*Function*

### Purpose

Destroy a message.

### C Declaration

```
tibems_status tibemsMsg_Destroy(
    tibemsMsg message );
```

## COBOL Call

```
CALL "tibemsMsg_Destroy"
    USING BY VALUE message,
```

```
RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Destroy this message.

## tibemsMsg\_GetAsBytes

*Function*

### Purpose

Get a byte sequence representation of the message object.

### C Declaration

```
tibems_status tibemsMsg_GetAsBytes(
    const tibemsMsg message,
    const void** bytes,
    tibems_int* actual_size );
```

### COBOL Call

```
CALL "tibemsMsg_GetAsBytes"
    USING BY VALUE message,
         BY REFERENCE bytes,
         BY REFERENCE actual-size,
         RETURNING tibems-status
END-CALL.
```



message and bytes have usage pointer.

### Parameters

Parameter	Description
message	Fill the byte array with the content of this message.
bytes	Allocate a byte sequence, and store a pointer to it in this location.
actual_size	Store the length of the byte sequence at this location.

### Remarks

This call formats the data of the message as a byte sequence in EMS wire format, which is suitable for archiving in a file.

The function allocates storage for the byte sequence, and associates it with the message; the byte sequence storage persists until your program destroys the message object.

Your program *must not* modify the byte sequence. To make a modifiable byte sequence, use [tibemsMsg\\_GetAsBytesCopy](#) instead.

The byte sequence includes data from the message header, message properties, and all message fields.

The byte sequence might contain interior null bytes.

**See Also**[tibemsMsg\\_CreateFromBytes](#)[tibemsMsg\\_GetAsBytesCopy](#)[tibemsMsg\\_GetByteSize](#)**tibemsMsg\_GetAsBytesCopy***Function***Purpose**

Copy a byte sequence representation of the message object into storage supplied by the program.

**C Declaration**

```
tibems_status tibemsMsg_GetAsBytesCopy(
    const tibemsMsg message,
    const void* bytes,
    tibems_int avail_size,
    tibems_int* actual_size );
```

**COBOL Call**

```
CALL "tibemsMsg_GetAsBytesCopy"
    USING BY VALUE message,
          BY REFERENCE bytes,
          BY VALUE avail-size,
          BY REFERENCE actual-size,
          RETURNING tibems-status
END-CALL.
```



message has usage pointer.

**Parameters**

Parameter	Description
message	Fill the byte array with the content of this message.
bytes	Store the byte sequence in this location. Your program must supply suitable storage for a byte sequence.
avail_size	Length of the storage available for the byte sequence.
actual_size	Store the length of the byte sequence.

**Remarks**

This call formats the data of the message as a byte sequence in EMS wire format, which is suitable for archiving in a file.

Your program must allocate storage for the byte sequence, and supply a pointer to it as an argument.

The byte sequence includes data from the message header, message properties, and all message fields.

The byte sequence might contain interior null bytes.

Status Code	Description
<a href="#">TIBEMS_INSUFFICIENT_BUFFER</a>	The buffer is not large enough for the data. The return parameter <code>actual_size</code> indicates the size of the required buffer.

### See Also

[tibemsMsg\\_CreateFromBytes](#)

[tibemsMsg\\_GetAsBytes](#)

[tibemsMsg\\_GetByteSize](#)

## tibemsMsg\_GetBodyType

*Function*

### Purpose

Get the body type of a message.



Message body type is distinct from message type—even though they have similar names. Contrast [tibemsMsg\\_GetType](#).

### C Declaration

```
tibems_status tibemsMsg_GetBodyType(
    tibemsMsg message,
    tibemsMsgType* type );
```

### COBOL Call

```
CALL "tibemsMsg_GetBodyType"
    USING BY VALUE message,
         BY REFERENCE type,
         RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Get the body type of this message.
type	Store the body type.

### See Also

[Body Types](#)

[tibemsMsgType](#)

## tibemsMsg\_GetByteSize

Function

### Purpose

Compute the size of the byte sequence representation of the message.

### C Declaration

```
tibems_status tibemsMsg_GetByteSize(
    tibemsMsg message,
    tibems_int* size );
```

### COBOL Call

```
CALL "tibemsMsg_GetByteSize"
    USING BY VALUE message,
         BY REFERENCE size,
         RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Compute the size of this message.
size	Store the message size.

### Remarks

This call computes the size of a message in bytes. This measurement accounts for the actual space that the wire format message occupies, including its header, properties, and body data. (It does not include allocated storage that remains unused.)

Before calling [tibemsMsg\\_GetAsBytesCopy](#), use this call to measure the size of a message, then allocate sufficient space to store a copy.

You can also use this call to measure network throughput, or to limit a program's output rate (also called *throttling*).

### Deprecated Form

```
tibems_int tibemsMsg_ByteSize(
    tibemsMsg message );
```



Do not use this deprecated form. It will become obsolete in a future release.

### See Also

[tibemsMsg\\_GetAsBytesCopy](#)

## tibemsMsg\_GetCorrelationID

Function

### Purpose

Get the correlation ID header of a message.

### C Declaration

```
tibems_status tibemsMsg_GetCorrelationID(
    tibemsMsg message,
    const char** value );
```

### COBOL Call

```
CALL "tibemsMsg_GetCorrelationID"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.
```



message and value have usage pointer.

### Parameters

Parameter	Description
message	Get the correlation ID of this message.
value	Store the correlation ID.

### Remarks

Correlation ID refers to a related message. For example, when a consumer responds to a request message by sending a reply, it can set the correlation ID of the reply to indicate the request message.

The JMS specification allows three categories of values for the correlation ID property:

- **Message ID**

A message ID is a unique string that the provider assigns to a message. Programs can use these IDs to correlate messages. For example, a program can link a response to a request by setting the correlation ID of a response message to the message ID of the corresponding request message.

Message ID strings begin with the prefix ID: (which is reserved for this purpose).

- **String**

Programs can also correlate messages using arbitrary strings, with semantics determined by the application.

These strings must *not* begin with the prefix ID: (which is reserved for message IDs).

- **Byte Array**

This implementation does not support byte array values for the correlation ID property. The JMS specification does not require support.

### See Also

[tibemsMsg\\_GetMessageID](#)

[tibemsMsg\\_SetCorrelationID](#)

## tibemsMsg\_GetDeliveryMode

*Function*

### Purpose

Get the delivery mode header from a message.

### C Declaration

```
tibems_status tibemsMsg_GetDeliveryMode(
    tibemsMsg message,
    tibemsDeliveryMode* value );
```

### COBOL Call

```
CALL "tibemsMsg_GetDeliveryMode"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Get the delivery mode from this message.
value	Store the delivery mode.

### Remarks

Delivery mode is a header property of message objects.

### See Also

[tibemsDeliveryMode](#)

[tibemsMsg\\_SetDeliveryMode](#)

## tibemsMsg\_GetDeliveryTime

*Function*

### Purpose

Get the delivery time header from a message.

### C Declaration

```
tibems_status tibemsMsg_GetDeliveryTime(
    tibemsMsg message,
    tibems_long* time );
```

### COBOL Call

```
CALL "tibemsMsg_GetDeliveryTime"
    USING BY VALUE message,
         BY REFERENCE time,
```

RETURNING tibems-status  
END-CALL.



message has usage pointer.

### Parameters

Parameter	Description
message	Get the delivery time from this message.
time	Store the delivery time.

### Remarks

Sending calls record the delivery time of the message in this field.

The delivery time is the difference, measured in milliseconds, between the delivery time and midnight, January 1, 1970 UTC.

A message's delivery time is the earliest time when a JMS provider may deliver the message to a consumer. The provider must not deliver messages before the delivery time has been reached.

### See Also

[tibemsMsgProducer\\_GetDeliveryDelay](#)

[tibemsMsgProducer\\_SetDeliveryDelay](#)

## tibemsMsg\_GetDestination

*Function*

### Purpose

Get the destination header from a message.

### C Declaration

```
tibems_status tibemsMsg_GetDestination(
    tibemsMsg message,
    tibemsDestination* value );
```

### COBOL Call

```
CALL "tibemsMsg_GetDestination"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.
```



message and value have usage pointer.

### Parameters

Parameter	Description
message	Get the destination from this message.
value	Store the destination.

## Remarks

Sending calls record the destination (queue or topic) of the message in this header (ignoring and overwriting any existing value). The value is based on either a property of the producer, or on a parameter to the send call.

Listeners that consume messages from several destinations can use this property to determine the actual destination of a message.

## See Also

[tibemsDestination](#)

[tibemsMsg\\_SetDestination](#)

## tibemsMsg\_GetEncoding

*Function*

### Purpose

Get the character encoding header from a message.

### C Declaration

```
tibems_status tibemsMsg_GetEncoding(
    const tibemsMsg message,
    const char** value );
```

### COBOL Call

```
CALL "tibemsMsg_GetEncoding"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.
```



message and value have usage pointer.

### Parameters

Parameter	Description
message	Get the character encoding from this message.
value	Store the character encoding.

### Remarks

This call extends the JMS specification.

This encoding applies to all strings in message bodies (names and values), and properties (names and values). It does *not* apply to header names nor values.

### See Also

[Strings and Character Encodings](#)

[tibemsMsg\\_SetEncoding](#)

## tibemsMsg\_GetExpiration

*Function*

### Purpose

Get the expiration header from a message.

### C Declaration

```
tibems_status tibemsMsg_GetExpiration(
    tibemsMsg message,
    tibems_long* value );
```

### COBOL Call

```
CALL "tibemsMsg_GetExpiration"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Get the expiration from this message.
value	Store the expiration.

### Remarks

Sending calls record the expiration time (in milliseconds) of the message in this field:

- If the time-to-live is non-zero, the expiration is the sum of that time-to-live and the sending client's current time (GMT).
- If the time-to-live is zero, then expiration is also zero—indicating that the message never expires. The server discards a message when its expiration time has passed. However, the JMS specification does not guarantee that clients do not receive expired messages.

### See Also

[tibemsMsg\\_SetExpiration](#)

[tibemsMsgProducer\\_GetTimeToLive](#)

## tibemsMsg\_GetMessageID

*Function*

### Purpose

Get the message ID header from a message.

### C Declaration

```
tibems_status tibemsMsg_GetMessageID(
    tibemsMsg message,
    const char** value );
```

## COBOL Call

```
CALL "tibemsMsg_GetMessageID"
    USING BY VALUE message,
          BY REFERENCE value,
          RETURNING tibems-status
END-CALL.
```



message and value have usage pointer.

## Parameters

Parameter	Description
message	Get the message ID from this message.
value	Store the message ID.

## Remarks

Sending calls assign a unique ID to each message, and record it in this header.

All message ID values start with the 3-character prefix ID: (which is reserved for this purpose).

Applications that do not require message IDs can reduce overhead costs by disabling IDs; see [tibemsMsgProducer\\_SetDisableMessageID](#). When the producer disables IDs, the value of this header is null.

## See Also

[tibemsMsg\\_GetCorrelationID](#)

[tibemsMsg\\_SetMessageID](#)

[tibemsMsgProducer\\_SetDisableMessageID](#)

## tibemsMsg\_GetPriority

*Function*

### Purpose

Get the priority header from a message.

### C Declaration

```
tibems_status tibemsMsg_GetPriority(
    tibemsMsg message,
    tibems_int* value );
```

## COBOL Call

```
CALL "tibemsMsg_GetPriority"
    USING BY VALUE message,
          BY REFERENCE value,
          RETURNING tibems-status
END-CALL.
```



message has usage pointer.

## Parameters

Parameter	Description
message	Get the priority from this message.
value	Store the priority.

## Remarks

Sending calls record the priority of a message in this header, based on either a property of the producer, or on a parameter to the send call.

The JMS specification defines ten levels of priority value, from zero (lowest priority) to 9 (highest priority). The specification suggests that clients consider 0–4 as gradations of normal priority, and priorities 5–9 as gradations of expedited priority.

Priority affects the order in which the server delivers messages to consumers (higher values first). The JMS specification does not require all providers to implement priority ordering of messages. (EMS supports priorities, but other JMS providers might not.)

## See Also

[tibemsMsg\\_SetPriority](#)

[tibemsMsgProducer\\_Send](#)

[tibemsMsgProducer\\_SetPriority](#)

## tibemsMsg\_Get Property

*Function*

### Purpose

Get the value of a message property.

### C Declaration

```
tibems_status tibemsMsg_Get Property(
    tibemsMsg message,
    const char* name,
    tibemsMsgField* value );

tibems_status tibemsMsg_GetBooleanProperty(
    tibemsMsg message,
    const char* name,
    tibems_bool* value );

tibems_status tibemsMsg_GetByteProperty(
    tibemsMsg message,
    const char* name,
    tibems_byte* value );

tibems_status tibemsMsg_GetDoubleProperty(
    tibemsMsg message,
    const char* name,
    tibems_double* value );

tibems_status tibemsMsg_GetFloatProperty(
    tibemsMsg message,
    const char* name,
    tibems_float* value );
```

```

tibems_status tibemsMsg_GetIntProperty(
    tibemsMsg message,
    const char* name,
    tibems_int* value );

tibems_status tibemsMsg_GetLongProperty(
    tibemsMsg message,
    const char* name,
    tibems_long* value );

tibems_status tibemsMsg_GetShortProperty(
    tibemsMsg message,
    const char* name,
    tibems_short* value );

tibems_status tibemsMsg_GetStringProperty(
    tibemsMsg message,
    const char* name,
    char** value );

```

### COBOL Call

```

CALL "tibemsMsg_Get_Property"
    USING BY VALUE message,
         BY REFERENCE name,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsMsg_GetBooleanProperty"
    USING BY VALUE message,
         BY REFERENCE name,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsMsg_GetByteProperty"
    USING BY VALUE message,
         BY REFERENCE name,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsMsg_GetDoubleProperty"
    USING BY VALUE message,
         BY REFERENCE name,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsMsg_GetFloatProperty"
    USING BY VALUE message,
         BY REFERENCE name,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsMsg_GetIntProperty"
    USING BY VALUE message,
         BY REFERENCE name,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsMsg_GetLongProperty"
    USING BY VALUE message,
         BY REFERENCE name,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsMsg_GetShortProperty"

```

```

        USING BY VALUE message,
              BY REFERENCE name,
              BY REFERENCE value,
              RETURNING tibems-status
END-CALL.

CALL "tibemsMsg_GetStringProperty"
    USING BY VALUE message,
          BY REFERENCE name,
          BY REFERENCE value,
          RETURNING tibems-status
END-CALL.

```



message has usage pointer.

value has usage pointer only in `tibemsMsg_GetStringProperty` (but not in the other calls documented in this group).

### Parameters

Parameter	Description
message	Get a property from this message.
name	Get the property with this name (case sensitive). Property names must obey the JMS rules for a message selector identifier (see <a href="#">Message Selectors</a> ). Property names must not be null, and must not be empty strings.
value	Store the value.

### Remarks

The JMS specification defines eight calls to get properties with different value types—converting between compatible types. All of these functions convert property values to the corresponding type (if possible).

Status Code	Description
<a href="#">TIBEMS_CONVERSION_FAILED</a>	The actual type of the property is not compatible with the requested type.
<a href="#">TIBEMS_NOT_FOUND</a>	The property is not set in this message.

### See Also

[tibemsMsg\\_Set Property](#)

## **tibemsMsg\_GetPropertyNames**

*Function*

### Purpose

Get a list of property names from a message.

## C Declaration

```
tibems_status tibemsMsg_GetPropertyNames(
    tibemsMsg message,
    tibemsMsgEnum* enumeration );
```

## COBOL Call

```
CALL "tibemsMsg_GetPropertyNames"
    USING BY VALUE message,
         BY REFERENCE enumeration,
         RETURNING tibems-status
END-CALL.
```



message and enumeration have usage pointer.

## Parameters

Parameter	Description
message	Get the property names from this message.
enumeration	Store the property names.

## See Also

[tibemsMsgEnum](#)

## tibemsMsg\_GetRedelivered

*Function*

### Purpose

Get the redelivered header from a message.

## C Declaration

```
tibems_status tibemsMsg_GetRedelivered(
    tibemsMsg message,
    tibems_bool* value );
```

## COBOL Call

```
CALL "tibemsMsg_GetRedelivered"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.
```



message has usage pointer.

## Parameters

Parameter	Description
message	Get the redelivered indicator from this message.
value	Store the redelivered indicator in this location.

## Remarks

The server sets this header to indicate whether a message might duplicate a previously delivered message:

- false—The server has *not* previously attempted to deliver this message to the consumer.
- true—It is likely, but not guaranteed, that this message was delivered earlier but that its receipt was not acknowledged at that time.

## See Also

[tibemsMsg\\_SetRedelivered](#)

## tibemsMsg\_GetReplyTo

*Function*

### Purpose

Get the reply-to header from a message.

### C Declaration

```
tibems_status tibemsMsg_GetReplyTo(
    tibemsMsg message,
    tibemsDestination* value );
```

### COBOL Call

```
CALL "tibemsMsg_GetReplyTo"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.
```



message and value have usage pointer.

### Parameters

Parameter	Description
message	Get the reply-to header from this message.
value	Store the reply-to header.

### Remarks

Sending clients can set this header to request that recipients reply to the message:

- When the value is a destination object, recipients can send replies to that destination. Such a message is called a *request*.
- When the value is null, the sender does not expect a reply.

When sending a reply, clients can refer to the corresponding request by setting the correlation ID field.

### See Also

[tibemsMsg\\_SetCorrelationID](#)

[tibemsMsg\\_SetReplyTo](#)

## tibemsMsg\_GetTimestamp

*Function*

### Purpose

Get the timestamp header from a message.

### C Declaration

```
tibems_status tibemsMsg_GetTimestamp(
    tibemsMsg message,
    tibems_long* value );
```

### COBOL Call

```
CALL "tibemsMsg_GetTimestamp"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Get the timestamp from this message.
value	Store the timestamp.

### Remarks

Sending calls record a UTC timestamp in this header, indicating the approximate time that the server accepted the message.

The value is in milliseconds since January 1, 1970 (as in Java).

Applications that do not require timestamps can reduce overhead costs by disabling timestamps; see [tibemsMsgProducer\\_SetDisableMessageTimestamp](#). When the producer disables timestamps, the value of this header is zero.

### See Also

[tibemsMsg\\_SetTimestamp](#)

[tibemsMsgProducer\\_SetDisableMessageTimestamp](#)

## tibemsMsg\_GetType

*Function*

### Purpose

Get the type header of a message.



Message type is distinct from message body type—even though they have similar names. Contrast [tibemsMsg\\_GetBodyType](#).

## C Declaration

```
tibems_status tibemsMsg_GetType(
    tibemsMsg message,
    const char** value );
```

## COBOL Call

```
CALL "tibemsMsg_GetType"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.
```



message and value have usage pointer.

## Parameters

Parameter	Description
message	Get the type header of this message.
value	Store the type.

## Remarks

Some JMS providers use a message repository to store message type definitions. Client programs can store a body type that references a definition in the repository. EMS supports this header, but does not use it.

The JMS specification does not define a standard message definition repository, nor does it define a naming policy for message type definitions.

Some providers require message type definitions for each application message. To ensure compatibility with such providers, client programs can set this header, even if the client application does not use it.

To ensure portability, clients can set this header with symbolic values (rather than literals), and configure them to match the provider's repository.

## See Also

[tibemsMsg\\_SetType](#)

## tibemsMsg\_MakeWriteable

*Function*

### Purpose

Make a message writeable.

## C Declaration

```
tibems_status tibemsMsg_MakeWriteable(
    tibemsMsg message );
```

## COBOL Call

```
CALL "tibemsMsg_MakeWriteable"
    USING BY VALUE message,
         RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Make this message writeable.

## tibemsMsg\_Print

*Function*

### Purpose

Print a message.

### C Declaration

```
void tibemsMsg_Print(
    tibemsMsg message );
```

### COBOL Call

```
CALL "tibemsMsg_Print"
    USING BY VALUE message,
    END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Print this message.

### Remarks

This call prints a string that includes the body type, headers (name-value pairs), properties (name-value pairs), and body content.

tibemsMsg\_Print prints the message to stdout.

### See Also

[tibemsMsg\\_PrintToBuffer](#)

## tibemsMsg\_PrintToBuffer

*Function*

### Purpose

Print a message into a buffer.

### C Declaration

```
tibems_status tibemsMsg_PrintToBuffer(
    tibemsMsg message,
    char* buffer,
    tibems_int maxlen );
```

## COBOL Call

```
CALL "tibemsMsg_PrintToBuffer"
  USING BY VALUE message,
  BY REFERENCE buffer,
  BY VALUE maxlen,
  RETURNING tibems-status
END-CALL.
```



message has usage pointer.

## Parameters

Parameter	Description
message	Print this message.
buffer	Location to store the string representation of the message.
maxlen	Size of the buffer.

## Remarks

This call prints a string that includes the body type, headers (name-value pairs), properties (name-value pairs), and body content.

`tibemsMsg_PrintToBuffer` prints the message to a buffer.

Status Code	Description
TIBEMS_INSUFFICIENT_BUFFER	The buffer is too small to hold the data.

## See Also

[tibemsMsg\\_Print](#)

## tibemsMsg\_PropertyExists

*Function*

### Purpose

Test whether a named property has been set on a message.

### C Declaration

```
tibems_status tibemsMsg_PropertyExists(
  tibemsMsg message,
  const char* name,
  tibems_bool* result );
```

## COBOL Call

```
CALL "tibemsMsg_PropertyExists"
  USING BY VALUE message,
  BY REFERENCE name,
  BY REFERENCE result,
  RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Test this message for the property.
name	Test whether the message has a property with this name.
result	Store the boolean result of the test in this location: <ul style="list-style-type: none"> <li>TIBEMS_TRUE if the property has a value on the message</li> <li>TIBEMS_FALSE otherwise</li> </ul>

## tibemsMsg\_Recover

*Function*

### Purpose

Recover a single message.

### C Declaration

```
tibems_status tibemsMsg_Recover(
    tibemsMsg message );
```

### COBOL Call

```
CALL "tibemsMsg_Recover"
    USING BY VALUE message,
         RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Recover the specified message.

### Remarks

When the application calls this function, TIBCO Enterprise Message Service puts this message back on the queue or topic and makes it available for redelivery. The `JMSRedelivered` header is set to true.

This call is only legal for the `TIBEMS_EXPLICIT_CLIENT_ACKNOWLEDGE` and `TIBEMS_EXPLICIT_CLIENT_DUPS_OK_ACKNOWLEDGE` acknowledgment modes. In all other modes, this call returns `TIBEMS_ILLEGAL_STATE`.

This function recovers only the specified message. To recover all unacknowledged messages from a session, use `tibemsSession_Recover`.

## Restriction

It is illegal to call this function on an acknowledged message or on a message that is included in a transaction.

Additionally, it is illegal to call this function twice on the same message, or again on a message that was previously recovered using `tibemsSession_Recover`.

## See Also

[tibemsSession\\_Recover](#)

[tibemsAcknowledgeMode](#)

## tibemsMsg\_SetCorrelationID

*Function*

### Purpose

Set the correlation ID header of a message.

### C Declaration

```
tibems_status tibemsMsg_SetCorrelationID(
    tibemsMsg message,
    const char* value );
```

### COBOL Call

```
CALL "tibemsMsg_SetCorrelationID"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Set the correlation ID of this message.
value	Set the correlation ID to this value.

### Remarks

Correlation ID refers to a related message. For example, when a consumer responds to a request message by sending a reply, it can set the correlation ID of the reply to indicate the request message.

The JMS specification allows three categories of values for the correlation ID property:

- **Message ID**

A message ID is a unique string that the provider assigns to a message. Programs can use these IDs to correlate messages. For example, a program can link a response to a request by setting the correlation ID of a response message to the message ID of the corresponding request message.

Message ID strings begin with the prefix ID: (which is reserved for this purpose).

- **String**

Programs can also correlate messages using arbitrary strings, with semantics determined by the application.

These strings must *not* begin with the prefix ID: (which is reserved for message IDs).

- **Byte Array**

This implementation does not support byte array values for the correlation ID property. The JMS specification does not require support.

### See Also

[tibemsMsg\\_GetCorrelationID](#)

[tibemsMsg\\_GetMessageID](#)

## tibemsMsg\_SetDeliveryMode

*Function*

### Purpose

Set the delivery mode header of a message.

### C Declaration

```
tibems_status tibemsMsg_SetDeliveryMode(
    tibemsMsg message,
    tibemsDeliveryMode value );
```

### COBOL Call

```
CALL "tibemsMsg_SetDeliveryMode"
    USING BY VALUE message,
         BY VALUE value,
         RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Set the delivery mode of this message.
value	Set the delivery mode to this value.

### Remarks

Sending calls set the delivery mode header automatically. The JMS specification defines this call for symmetry.

### See Also

[tibemsDeliveryMode](#)

[tibemsMsg\\_GetDeliveryMode](#)

## tibemsMsg\_SetDestination

*Function*

### Purpose

Set the destination header of a message.

### C Declaration

```
tibems_status tibemsMsg_SetDestination(
    tibemsMsg message,
    tibemsDestination value );
```

### COBOL Call

```
CALL "tibemsMsg_SetDestination"
    USING BY VALUE message,
         BY VALUE value,
         RETURNING tibems-status
END-CALL.
```



message and value have usage pointer.

### Parameters

Parameter	Description
message	Set the destination of this message.
value	Set the destination to this value.

### Remarks

Sending calls set the delivery mode header automatically. The JMS specification defines this call for symmetry.

Sending calls record the destination (queue or topic) of the message in this header (ignoring and overwriting any existing value). The value is based on either a property of the producer, or on a parameter to the send call.

### See Also

[tibemsDestination](#)

[tibemsMsg\\_GetDestination](#)

[tibemsMsgProducer\\_Send](#)

[tibemsMsgProducer\\_GetDestination](#)

## tibemsMsg\_SetEncoding

*Function*

### Purpose

Set the character encoding header of a message.

## C Declaration

```
tibems_status tibemsMsg_SetEncoding(
    tibemsMsg message,
    const char* value );
```

## COBOL Call

```
CALL "tibemsMsg_SetEncoding"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.
```



message has usage pointer.

## Parameters

Parameter	Description
message	Indicate the character encoding of this message.
value	Indicate that the character encoding is this value.

## Remarks

This call extends the JMS specification.

Programs can indicate the encoding for individual messages.

`tibemsMsg_SetEncoding` does not actually do any encoding of message strings. An application uses this function to indicate the actual encoding of strings in the message.

When receiving a message in an EMS C client application, the encoding can be retrieved using [tibemsMsg\\_GetEncoding](#) and then use a third party library to do the actual decoding.

## See Also

[Strings and Character Encodings](#)

[tibemsMsg\\_GetEncoding](#)

## tibemsMsg\_SetExpiration

*Function*

### Purpose

Set the expiration header of a message.

## C Declaration

```
tibems_status tibemsMsg_SetExpiration(
    tibemsMsg message,
    tibems_long value );
```

## COBOL Call

```
CALL "tibemsMsg_SetExpiration"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Set the expiration of this message.
value	Set the expiration to this value.

### Remarks

Sending calls set the expiration header automatically. The JMS specification defines this call for symmetry.

Sending calls record the expiration time (in milliseconds) of the message in this field:

- If the time-to-live is non-zero, the expiration is the sum of that time-to-live and the sending client's current time (GMT).
- If the time-to-live is zero, then expiration is also zero—indicating that the message never expires.

The server discards a message when its expiration time has passed. However, the JMS specification does not guarantee that clients do not receive expired messages.

### See Also

[tibemsMsg\\_GetExpiration](#)

[tibemsMsgProducer\\_GetTimeToLive](#)

[tibemsMsgProducer\\_SetTimeToLive](#)

## tibemsMsg\_SetMessageID

*Function*

### Purpose

Set the message ID header of a message.

### C Declaration

```
tibems_status tibemsMsg_SetMessageID(
    tibemsMsg message,
    const char* value );
```

### COBOL Call

```
CALL "tibemsMsg_SetMessageID"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.
```



message has usage pointer.

## Parameters

Parameter	Description
message	Set the message ID of this message.
value	Set the message ID to this value.

## Remarks

Sending calls set the message ID header automatically. The JMS specification defines this call for symmetry.

Sending calls assign a unique ID to each message, and record it in this header.

All message ID values start with the 3-character prefix ID: (which is reserved for this purpose).

Applications that do not require message IDs can reduce overhead costs by disabling IDs; see [tibemsMsgProducer\\_SetDisableMessageID](#). When the producer disables IDs, the value of this header is null.

## See Also

[tibemsMsg\\_GetCorrelationID](#)

[tibemsMsg\\_GetMessageID](#)

[tibemsMsg\\_SetCorrelationID](#)

[tibemsMsgProducer\\_SetDisableMessageID](#)

## tibemsMsg\_SetPriority

*Function*

### Purpose

Set the priority header of a message.

### C Declaration

```
tibems_status tibemsMsg_SetPriority(
    tibemsMsg message,
    tibems_int value );
```

### COBOL Call

```
CALL "tibemsMsg_SetPriority"
    USING BY VALUE message,
         BY VALUE value,
         RETURNING tibems-status
END-CALL.
```



message has usage pointer.

## Parameters

Parameter	Description
message	Set the priority of this message.

Parameter	Description
value	Set the priority to this value.

### Remarks

Sending calls set the priority header automatically. The JMS specification defines this call for symmetry. Sending calls record the priority of a message in this header, based on either a property of the producer, or on a parameter to the send call.

The JMS specification defines ten levels of priority value, from zero (lowest priority) to 9 (highest priority). The specification suggests that clients consider 0–4 as gradations of normal priority, and priorities 5–9 as gradations of expedited priority.

Priority affects the order in which the server delivers messages to consumers (higher values first). The JMS specification does not require all providers to implement priority ordering of messages. (EMS supports priorities, but other JMS providers might not.)

### See Also

[tibemsMsg\\_GetPriority](#)

[tibemsMsgProducer\\_GetPriority](#)

[tibemsMsgProducer\\_SetPriority](#)

## tibemsMsg\_Set Property

*Function*

### Purpose

Set the value of a message property.

### C Declaration

```

tibems_status tibemsMsg_SetBooleanProperty(
    tibemsMsg message,
    const char* name,
    tibems_bool value );

tibems_status tibemsMsg_SetByteProperty(
    tibemsMsg message,
    const char* name,
    tibems_byte value );

tibems_status tibemsMsg_SetDoubleProperty(
    tibemsMsg message,
    const char* name,
    tibems_double value );

tibems_status tibemsMsg_SetFloatProperty(
    tibemsMsg message,
    const char* name,
    tibems_float value );

tibems_status tibemsMsg_SetIntProperty(
    tibemsMsg message,
    const char* name,
    tibems_int value );

tibems_status tibemsMsg_SetLongProperty(
    tibemsMsg message,
    const char* name,
    tibems_long value );

```

```

tibems_status tibemsMsg_SetShortProperty(
    tibemsMsg message,
    const char* name,
    tibems_short value );

tibems_status tibemsMsg_SetStringProperty(
    tibemsMsg message,
    const char* name,
    const char* value );

```

## COBOL Call

```

CALL "tibemsMsg_SetBooleanProperty"
    USING BY VALUE message,
         BY REFERENCE name,
         BY VALUE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsMsg_SetByteProperty"
    USING BY VALUE message,
         BY REFERENCE name,
         BY VALUE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsMsg_SetDoubleProperty"
    USING BY VALUE message,
         BY REFERENCE name,
         BY VALUE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsMsg_SetFloatProperty"
    USING BY VALUE message,
         BY REFERENCE name,
         BY VALUE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsMsg_SetIntProperty"
    USING BY VALUE message,
         BY REFERENCE name,
         BY VALUE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsMsg_SetLongProperty"
    USING BY VALUE message,
         BY REFERENCE name,
         BY VALUE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsMsg_SetShortProperty"
    USING BY VALUE message,
         BY REFERENCE name,
         BY VALUE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsMsg_SetStringProperty"
    USING BY VALUE message,
         BY REFERENCE name,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

```



message has usage pointer.

value has usage pointer only in `tibemsMsg_SetStringProperty` (but not in the other calls documented in this group).

### Parameters

Parameter	Description
message	Set a property on this message.  Property names must obey the JMS rules for a message selector identifier (see <a href="#">Message Selectors</a> ). Property names must not be null, and must not be empty strings.
name	Set a property with this name.  Property names must obey the JMS rules for a message selector identifier (see <a href="#">Message Selectors</a> ). Property names must not be null, and must not be empty strings.
value	Set the property to this value.

### Remarks

The JMS specification defines eight calls to set properties with different primitive value types.

Status Code	Description
<a href="#">TIBEMS_MSG_NOT_WRITEABLE</a>	The message is read-only.

### See Also

[tibemsMsg\\_Get Property](#)

## tibemsMsg\_SetRedelivered

*Function*

### Purpose

Set the redelivered header of a message.

### C Declaration

```
tibems_status tibemsMsg_SetRedelivered(
    tibemsMsg message,
    tibems_bool value );
```

### COBOL Call

```
CALL "tibemsMsg_SetRedelivered"
    USING BY VALUE message,
         BY VALUE value,
         RETURNING tibems-status
END-CALL.
```



message has usage pointer.

## Parameters

Parameter	Description
message	Set the redelivered indicator of this message.
value	Set the redelivered indicator to this value.

## Remarks

Sending calls set the redelivered header automatically. The JMS specification defines this call for symmetry.

The server sets this header to indicate whether a message might duplicate a previously delivered message:

- false—The server has *not* previously attempted to deliver this message to the consumer.
- true—It is likely (but not guaranteed) that the server has previously attempted to deliver this message to the consumer, but the consumer did not return timely acknowledgment.

## See Also

[tibemsMsg\\_GetRedelivered](#)

## tibemsMsg\_SetReplyTo

*Function*

### Purpose

Set the reply-to header of a message.

### C Declaration

```
tibems_status tibemsMsg_SetReplyTo(
    tibemsMsg message,
    tibemsDestination value );
```

### COBOL Call

```
CALL "tibemsMsg_SetReplyTo"
    USING BY VALUE message,
         BY VALUE value,
         RETURNING tibems-status
END-CALL.
```



message and value have usage pointer.

## Parameters

Parameter	Description
message	Set the reply-to header of this message.
value	Set the reply-to header to this value.

## Remarks

Sending clients can set this header to request that recipients reply to the message:

- When the value is a destination object, recipients can send replies to that destination. Such a message is called a *request*.
- When the value is null, the sender does not expect a reply.

When sending a reply, clients can refer to the corresponding request by setting the correlation ID field.

## See Also

[tibemsMsg\\_GetReplyTo](#)

[tibemsMsg\\_SetCorrelationID](#)

## tibemsMsg\_SetTimestamp

*Function*

### Purpose

Set the timestamp header of a message.

### C Declaration

```
tibems_status tibemsMsg_SetTimestamp(
    tibemsMsg message,
    tibems_long value );
```

### COBOL Call

```
CALL "tibemsMsg_SetTimestamp"
    USING BY VALUE message,
         BY VALUE value,
         RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Set the timestamp of this message.
value	Set the timestamp to this value.

## Remarks

Sending calls set the timestamp header automatically. The JMS specification defines this call for symmetry.

Sending calls record a UTC timestamp in this header, indicating the approximate time that the server accepted the message.

The value is in milliseconds since January 1, 1970 (as in Java).

Applications that do not require timestamps can reduce overhead costs by disabling timestamps; see [tibemsMsgProducer\\_SetDisableMessageTimestamp](#). When the producer disables timestamps, the value of this header is zero.

**See Also**[tibemsMsg\\_GetTimestamp](#)[tibemsMsgProducer\\_SetDisableMessageTimestamp](#)**tibemsMsg\_SetType***Function***Purpose**

Set the type header of a message.



Message type is distinct from message body type—even though they have similar names. Contrast [tibemsMsg\\_GetBodyType](#).

**C Declaration**

```
tibems_status tibemsMsg_SetType(
    tibemsMsg message,
    const char* value );
```

**COBOL Call**

```
CALL "tibemsMsg_SetType"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.
```



message has usage pointer.

**Parameters**

Parameter	Description
message	Set the type header of this message.
value	Set the type to this value.

**Remarks**

Some JMS providers use a message repository to store message type definitions. Client programs can store a body type that references a definition in the repository. EMS supports this header, but does not use it.

The JMS specification does not define a standard message definition repository, nor does it define a naming policy for message type definitions.

Some providers require message type definitions for each application message. To ensure compatibility with such providers, client programs can set this header, even if the client application does not use it.

To ensure portability, clients can set this header with symbolic values (rather than literals), and configure them to match the provider's repository.

**See Also**[tibemsMsg\\_GetType](#)

## tibemsBytesMsg

*Type*

### Purpose

Message containing a stream of uninterrupted bytes.

### Related Types

[tibemsMsg](#)

### Remarks

Messages with this body type contain a single value, which is a byte sequence.

Function	Description
<a href="#">tibemsBytesMsg_Create</a>	Create a bytes message.
<a href="#">tibemsBytesMsg_GetBodyLength</a>	Get the body length (in bytes) of a bytes message.
<a href="#">tibemsBytesMsg_GetBytes</a>	Get the body data of a bytes message.
<a href="#">tibemsBytesMsg_Read</a>	Read primitive datatypes from the byte stream in the message body.
<a href="#">tibemsBytesMsg_ReadBytes</a>	Read bytes to a byte sequence from the byte stream in the message body.
<a href="#">tibemsBytesMsg_Reset</a>	Set the read position to the beginning of the byte stream, and mark the message body as read-only.
<a href="#">tibemsBytesMsg_SetBytes</a>	Set the body data of a bytes message from a byte sequence.
<a href="#">tibemsBytesMsg_Write</a>	Write primitive datatypes to the byte stream in the message body.
<a href="#">tibemsBytesMsg_WriteBytes</a>	Write bytes from a byte array to the byte stream in the message body.

## tibemsBytesMsg\_Create

*Function*

### Purpose

Create a bytes message.

### C Declaration

```
tibems_status tibemsBytesMsg_Create(
    tibemsBytesMsg* message );
```

### COBOL Call

```
CALL "tibemsBytesMsg_Create"
    USING BY REFERENCE message,
```

RETURNING tibems-status  
END-CALL.



message has usage pointer.

### Parameters

Parameter	Description
message	Store a pointer to the new message.

### Remarks

This call creates a new bytes message.

When your application creates a message, it also allocates storage for that message. This storage must subsequently be freed by a call to `tibemsMsg_Destroy`.

### See Also

[tibemsMsg\\_Create](#)

[tibemsMsg\\_Destroy](#)

## tibemsBytesMsg\_GetBodyLength

*Function*

### Purpose

Get the body length (in bytes) of a bytes message.

### C Declaration

```
tibems_status tibemsBytesMsg_GetBodyLength(
    tibemsMsg message,
    tibems_int* return_length );
```

### COBOL Call

```
CALL "tibemsBytesMsg_GetBodyLength"
  USING BY VALUE message,
        BY REFERENCE return-length,
        RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Get the body length of this message.
return_length	Store the body length.

## tibemsBytesMsg\_GetBytes

*Function*

### Purpose

Get the body data of a bytes message.

### C Declaration

```
tibems_status tibemsBytesMsg_GetBytes(
    tibemsBytesMsg message,
    void** bytes,
    tibems_uint* byteSize );
```

### COBOL Call

```
CALL "tibemsBytesMsg_GetBytes"
  USING BY VALUE message,
        BY REFERENCE bytes,
        BY REFERENCE byteSize,
        RETURNING tibems-status
END-CALL.
```



message and bytes have usage pointer.

### Parameters

Parameter	Description
message	Get the byte sequence of this bytes message.
bytes	Store a pointer to the bytes of the message in this location. Your program must not change the bytes, which belong to the message; if you must modify the bytes, make a private copy first.
byteSize	Store the length of the byte sequence.

### Remarks

This call extracts a pointer to the body data of a bytes message.

The byte sequence storage persists until your program destroys the message object.

## tibemsBytesMsg\_Read

*Function*

### Purpose

Read primitive datatypes from the byte stream in the message body.

### C Declaration

```
tibems_status tibemsBytesMsg_ReadBoolean(
    tibemsBytesMsg message,
    tibems_bool* value );

tibems_status tibemsBytesMsg_ReadByte(
    tibemsBytesMsg message,
    tibems_byte* value );
```

```

tibems_status tibemsBytesMsg_ReadChar(
    tibemsBytesMsg message,
    tibems_wchar* value );

tibems_status tibemsBytesMsg_ReadDouble(
    tibemsBytesMsg message,
    tibems_double* value );

tibems_status tibemsBytesMsg_ReadFloat(
    tibemsBytesMsg message,
    tibems_float* value );

tibems_status tibemsBytesMsg_ReadInt(
    tibemsBytesMsg message,
    tibems_int* value );

tibems_status tibemsBytesMsg_ReadLong(
    tibemsBytesMsg message,
    tibems_long* value );

tibems_status tibemsBytesMsg_ReadShort(
    tibemsBytesMsg message,
    tibems_short* value );

tibems_status tibemsBytesMsg_ReadUnsignedByte(
    tibemsBytesMsg message,
    tibems_int* value );

tibems_status tibemsBytesMsg_ReadUnsignedShort(
    tibemsBytesMsg message,
    tibems_int* value );

```

## COBOL Call

```

CALL "tibemsBytesMsg_ReadBoolean"
    USING BY VALUE message,
         BY REFERENCE tibems-Boolean,
         RETURNING tibems-status
END-CALL.

```

```

CALL "tibemsBytesMsg_ReadByte"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

```

```

CALL "tibemsBytesMsg_ReadChar"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

```

```

CALL "tibemsBytesMsg_ReadDouble"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

```

```

CALL "tibemsBytesMsg_ReadFloat"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

```

```

CALL "tibemsBytesMsg_ReadInt"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

```

```

CALL "tibemsBytesMsg_ReadLong"
  USING BY VALUE message,
        BY REFERENCE value,
        RETURNING tibems-status
END-CALL.

CALL "tibemsBytesMsg_ReadShort"
  USING BY VALUE message,
        BY REFERENCE value,
        RETURNING tibems-status
END-CALL.

CALL "tibemsBytesMsg_ReadUnsignedByte"
  USING BY VALUE message,
        BY REFERENCE value,
        RETURNING tibems-status
END-CALL.

CALL "tibemsBytesMsg_ReadUnsignedShort"
  USING BY VALUE message,
        BY REFERENCE value,
        RETURNING tibems-status
END-CALL.

```



message has usage pointer in all calls.

### Parameters

Parameter	Description
message	Read a datum from the body byte stream of this message.
value	Store the datum.

### Remarks

The JMS specification defines eleven calls to extract data from the byte stream body of a [tibemsBytesMsg](#). Each call reads a unit of data from the stream, and advances the read position so that the next read call gets the next datum.

Function	# Bytes	Interpret As
tibemsBytesMsg_ReadBoolean	1	<a href="#">tibems_bool</a>
tibemsBytesMsg_ReadByte	1	<a href="#">tibems_byte</a>
tibemsBytesMsg_ReadUnsignedByte	1	<a href="#">tibems_int</a>
tibemsBytesMsg_ReadShort	2	<a href="#">tibems_short</a>
tibemsBytesMsg_ReadUnsignedShort	2	<a href="#">tibems_int</a>
tibemsBytesMsg_ReadChar	2	<a href="#">tibems_wchar</a>
tibemsBytesMsg_ReadInt	4	<a href="#">tibems_int</a>
tibemsBytesMsg_ReadLong	8	<a href="#">tibems_long</a>
tibemsBytesMsg_ReadFloat	4	<a href="#">tibems_float</a>

Function	# Bytes	Interpret As
tibemsBytesMsg_ReadDouble	8	tibems_double

### See Also

[tibemsBytesMsg\\_ReadBytes](#)

## tibemsBytesMsg\_ReadBytes

*Function*

### Purpose

Read bytes to a byte sequence from the byte stream in the message body.

### C Declaration

```
tibems_status tibemsBytesMsg_ReadBytes(
    tibemsBytesMsg message,
    const void** value,
    tibems_int requested_length,
    tibems_int* return_length );
```

### COBOL Call

```
CALL "tibemsBytesMsg_ReadBytes"
  USING BY VALUE message,
        BY REFERENCE value,
        BY VALUE requested-length,
        BY REFERENCE return-length,
        RETURNING tibems-status
END-CALL.
```



message and value have usage pointer.

### Parameters

Parameter	Description
message	Read bytes from the body of this message.
value	The program supplies a location. In that location, this call stores a pointer to the next block of bytes within the bytes message.  Your program must not change the bytes, which belong to the message; if you must modify the bytes, make a private copy first.
requested_length	Read (at most) this number of bytes from the stream.  This argument must be greater than zero.
return_length	Store the actual number of bytes that it read. (If the number of bytes remaining in the message is less than the requested_length, then this location indicates that number of remaining bytes. Your program must not use bytes beyond this limit.)  When the function cannot read even one byte, it stores -1 in this location (and returns a successful status code).

**Remarks**

Each call reads bytes from the stream into the byte array, and advances the read position.

**tibemsBytesMsg\_Reset**

*Function*

**Purpose**

Set the read position to the beginning of the byte stream, and mark the message body as read-only.

**C Declaration**

```
tibems_status tibemsBytesMsg_Reset(
    tibemsBytesMsg message );
```

**COBOL Call**

```
CALL "tibemsBytesMsg_Reset"
  USING BY VALUE message,
  RETURNING tibems-status
END-CALL.
```



message has usage pointer.

**Parameters**

Parameter	Description
message	Reset the read position for this message.

**Remarks**

This call prepares a message body for reading, as if the message were newly received. Contrast [tibemsMsg\\_ClearBody](#), which clears a message body in preparation for writing, as if it were newly created.

**tibemsBytesMsg\_SetBytes**

*Function*

**Purpose**

Set the body data of a bytes message from a byte sequence.

**C Declaration**

```
tibems_status tibemsBytesMsg_SetBytes(
    tibemsBytesMsg message,
    const void* bytes,
    tibems_uint byteSize );
```

**COBOL Call**

```
CALL "tibemsBytesMsg_SetBytes"
  USING BY VALUE message,
  BY REFERENCE bytes,
  BY VALUE byteSize,
  RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Set the body data of this bytes message.
bytes	Copy this byte sequence into the message body.
byteSize	Copy this number of bytes into the message.

## tibemsBytesMsg\_Write

*Function*

### Purpose

Write primitive datatypes to the byte stream in the message body.

### C Declaration

```

tibems_status tibemsBytesMsg_WriteBoolean(
    tibemsBytesMsg message,
    tibems_bool value );

tibems_status tibemsBytesMsg_WriteByte(
    tibemsBytesMsg message,
    tibems_byte value );

tibems_status tibemsBytesMsg_WriteChar(
    tibemsBytesMsg message,
    tibems_wchar value );

tibems_status tibemsBytesMsg_WriteDouble(
    tibemsBytesMsg message,
    tibems_double value );

tibems_status tibemsBytesMsg_WriteFloat(
    tibemsBytesMsg message,
    tibems_float value );

tibems_status tibemsBytesMsg_WriteInt(
    tibemsBytesMsg message,
    tibems_int value );

tibems_status tibemsBytesMsg_WriteLong(
    tibemsBytesMsg message,
    tibems_long value );

tibems_status tibemsBytesMsg_WriteShort(
    tibemsBytesMsg message,
    tibems_short value );

```

### COBOL Call

```

CALL "tibemsBytesMsg_WriteBoolean"
  USING BY VALUE message,
        BY VALUE tibems-Boolean,
        RETURNING tibems-status
END-CALL.

```

```

CALL "tibemsBytesMsg_WriteByte"
  USING BY VALUE message,
        BY VALUE value,

```

```

        RETURNING tibems-status
    END-CALL.

    CALL "tibemsBytesMsg_WriteChar"
        USING BY VALUE message,
             BY VALUE value,
        RETURNING tibems-status
    END-CALL.

    CALL "tibemsBytesMsg_WriteDouble"
        USING BY VALUE message,
             BY VALUE value,
        RETURNING tibems-status
    END-CALL.

    CALL "tibemsBytesMsg_WriteFloat"
        USING BY VALUE message,
             BY VALUE value,
        RETURNING tibems-status
    END-CALL.

    CALL "tibemsBytesMsg_WriteInt"
        USING BY VALUE message,
             BY VALUE value,
        RETURNING tibems-status
    END-CALL.

    CALL "tibemsBytesMsg_WriteLong"
        USING BY VALUE message,
             BY VALUE value,
        RETURNING tibems-status
    END-CALL.

    CALL "tibemsBytesMsg_WriteShort"
        USING BY VALUE message,
             BY VALUE value,
        RETURNING tibems-status
    END-CALL.

```



message has usage pointer in all calls.

### Parameters

Parameter	Description
message	Write data to the body of this bytes message.
value	Write this value to the message.

### Remarks

The JMS specification defines these nine calls to insert data into the byte stream of a `BytesMessage`.

Each call writes a data value to the stream, and advances the write position so that the next write call appends to the new end of the stream.

Function	# Bytes	Notes
<code>tibemsBytesMsg_WriteBoolean</code>	1	This function writes true as (byte)1, and false as (byte)0.
<code>tibemsBytesMsg_WriteByte</code>	1	

Function	# Bytes	Notes
<code>tibemsBytesMsg_WriteShort</code>	2	
<code>tibemsBytesMsg_WriteChar</code>	2	This function writes a 2-byte character—high byte first.
<code>tibemsBytesMsg_WriteInt</code>	4	
<code>tibemsBytesMsg_WriteLong</code>	8	
<code>tibemsBytesMsg_WriteFloat</code>	4	
<code>tibemsBytesMsg_WriteDouble</code>	8	

### See Also

[tibemsBytesMsg\\_WriteBytes](#)

## tibemsBytesMsg\_WriteBytes

*Function*

### Purpose

Write bytes from a byte array to the byte stream in the message body.

### C Declaration

```
tibems_status tibemsBytesMsg_WriteBytes(
    tibemsBytesMsg message,
    const void* value,
    tibems_uint length );
```

### COBOL Call

```
CALL "tibemsBytesMsg_WriteBytes"
  USING BY VALUE message,
        BY REFERENCE value,
        BY VALUE size,
        RETURNING tibems-status
END-CALL.
```



message has usage pointer in all calls.

### Parameters

Parameter	Description
message	Write bytes into the body data of this bytes message.
value	Write bytes from this byte array into the message.
length	Write this number of bytes from the byte array.

### Remarks

Each call writes bytes from the byte array into the stream, and advances the write position.

## tibemsMapMsg

*Type*

### Purpose

A message containing a set of name-value pairs.

### Related Types

[tibemsMsg](#)

### Remarks

Messages with this body type contain several values, indexed by name.

Function	Description
<a href="#">tibemsMapMsg_Create</a>	Create a map message.
<a href="#">tibemsMapMsg_Get</a>	Get data values from a map message.
<a href="#">tibemsMapMsg_GetMapNames</a>	Get an enumeration of the field names in a map message.
<a href="#">tibemsMapMsg_ItemExists</a>	Test if a named pair exists.
<a href="#">tibemsMapMsg_Set</a>	Set a name-value pair in a map message.
<a href="#">tibemsMapMsg_SetBytes</a>	Set a byte array as a named value in a map message.

### Extensions

TIBCO Enterprise Message Service extends the JMS `MapMessage` and `StreamMessage` body types in two ways. These extensions allow TIBCO Enterprise Message Service to exchange messages with TIBCO Rendezvous and TIBCO SmartSockets programs, which have certain features not available within the JMS specification.

- You can insert another `MapMessage` or `StreamMessage` instance as a submessage into a `MapMessage` or `StreamMessage`, generating a series of nested messages, instead of a flat message.
- You can use arrays as well as primitive types for the values.

These extensions add considerable flexibility to the two body types. However, they are extensions and therefore not compliant with JMS specifications. Extended messages are tagged as extensions with the vendor property tag `JMS_TIBCO_MSG_EXT`.

For more information on message compatibility with Rendezvous messages, see "Message Body" in *TIBCO Enterprise Message Service Release Notes*.

## tibemsMapMsg\_Create

*Function*

### Purpose

Create a map message.

## C Declaration

```
tibems_status tibemsMapMsg_Create(
    tibemsMapMsg* message );
```

## COBOL Call

```
CALL "tibemsMapMsg_Create"
    USING BY REFERENCE message,
        RETURNING tibems-status
END-CALL.
```



message has usage pointer.

## Parameters

Parameter	Description
message	Store a pointer to the new message.

## Remarks

Create a new map message.

When your application creates a message, it also allocates storage for that message. This storage must subsequently be freed by a call to `tibemsMsg_Destroy`.

## See Also

[tibemsMsg\\_Create](#)

[tibemsMsg\\_Destroy](#)

## tibemsMapMsg\_Get

*Function*

### Purpose

Get data values from a map message.

## C Declaration

```
tibems_status tibemsMapMsg_GetBoolean(
    tibemsMapMsg message,
    const char* name,
    tibems_bool* value );
```

```
tibems_status tibemsMapMsg_GetByte(
    tibemsMapMsg message,
    const char* name,
    tibems_byte* value );
```

```
tibems_status tibemsMapMsg_GetBytes(
    tibemsMapMsg message,
    const char* name,
    void** bytes,
    tibems_uint* bytesSize );
```

```
tibems_status tibemsMapMsg_GetChar(
    tibemsMapMsg message,
    const char* name,
    tibems_wchar* value );
```

```

tibems_status tibemsMapMsg_GetDouble(
    tibemsMapMsg message,
    const char* name,
    tibems_double* value );

tibems_status tibemsMapMsg_GetField(
    tibemsMapMsg message,
    const char* name,
    tibemsMsgField* value );

tibems_status tibemsMapMsg_GetFloat(
    tibemsMapMsg message,
    const char* name,
    tibems_float* value );

tibems_status tibemsMapMsg_GetInt(
    tibemsMapMsg message,
    const char* name,
    tibems_int* value );

tibems_status tibemsMapMsg_GetLong(
    tibemsMapMsg message,
    const char* name,
    tibems_long* value );

tibems_status tibemsMapMsg_GetMapMsg(
    tibemsMapMsg message,
    const char* name,
    tibemsMapMsg* value );

tibems_status tibemsMapMsg_GetShort(
    tibemsMapMsg message,
    const char* name,
    tibems_short* value );

tibems_status tibemsMapMsg_GetString(
    tibemsMapMsg message,
    const char* name,
    const char** value );

```

## COBOL Call

```

CALL "tibemsMapMsg_GetBoolean"
    USING BY VALUE message,
          BY REFERENCE name,
          BY REFERENCE value,
          RETURNING tibems-status
END-CALL.

CALL "tibemsMapMsg_GetBytes"
    USING BY VALUE message,
          BY REFERENCE name,
          BY REFERENCE bytes,
          BY REFERENCE bytesSize,
          RETURNING tibems-status
END-CALL.

CALL "tibemsMapMsg_GetByte"
    USING BY VALUE message,
          BY REFERENCE name,
          BY REFERENCE value,
          RETURNING tibems-status
END-CALL.

CALL "tibemsMapMsg_GetChar"
    USING BY VALUE message,
          BY REFERENCE name,
          BY REFERENCE value,
          RETURNING tibems-status
END-CALL.

```

```

CALL "tibemsMapMsg_GetDouble"
  USING BY VALUE message,
        BY REFERENCE name,
        BY REFERENCE value,
        RETURNING tibems-status
END-CALL.

CALL "tibemsMapMsg_GetField"
  USING BY VALUE message,
        BY REFERENCE name,
        BY REFERENCE value,
        RETURNING tibems-status
END-CALL.

CALL "tibemsMapMsg_GetFloat"
  USING BY VALUE message,
        BY REFERENCE name,
        BY REFERENCE value,
        RETURNING tibems-status
END-CALL.

CALL "tibemsMapMsg_GetInt"
  USING BY VALUE message,
        BY REFERENCE name,
        BY REFERENCE value,
        RETURNING tibems-status
END-CALL.

CALL "tibemsMapMsg_GetLong"
  USING BY VALUE message,
        BY REFERENCE name,
        BY REFERENCE value,
        RETURNING tibems-status
END-CALL.

CALL "tibemsMapMsg_GetMapMsg"
  USING BY VALUE message,
        BY REFERENCE name,
        BY REFERENCE value,
        RETURNING tibems-status
END-CALL.

CALL "tibemsMapMsg_GetShort"
  USING BY VALUE message,
        BY REFERENCE name,
        BY REFERENCE value,
        RETURNING tibems-status
END-CALL.

CALL "tibemsMapMsg_GetString"
  USING BY VALUE message,
        BY REFERENCE name,
        BY REFERENCE value,
        RETURNING tibems-status
END-CALL.

```

message and bytes have usage pointer.

value has usage pointer only in `tibemsMapMsg_GetMapMsg` and `tibemsMapMsg_GetString` (but not in the other calls documented in this group).

### Parameters

Parameter	Description
message	Get a value from this map message.
name	Get the value associated with this name.

Parameter	Description
value	For unitary values, the function copies the value into this location. For strings, nested messages and fields, the function stores (in this location) a pointer to the value within the message.
bytes	tibemsMapMsg_GetBytes stores a pointer to the byte sequence in this location.
bytesSize	tibemsMapMsg_GetBytes stores the length of the byte sequence in this location.

### Remarks

The JMS specification defines these calls to extract data from the name-value pairs of a map message.

To get array values from a map message, call `tibemsMapMsg_GetField`, then extract the array value from the field; see [tibemsMsgField](#).

When the message does not have a field set for the name, these calls return `TIBEMS_NOT_FOUND`.

## tibemsMapMsg\_GetMapNames

*Function*

### Purpose

Get an enumeration of the field names in a map message.

### C Declaration

```
tibems_status tibemsMapMsg_GetMapNames(
    tibemsMsg message,
    tibemsMsgEnum* enumeration );
```

### COBOL Call

```
CALL "tibemsMapMsg_GetMapNames"
    USING BY VALUE message,
         BY REFERENCE enumeration,
         RETURNING tibems-status
END-CALL.
```



message and enumeration have usage pointer.

### Parameters

Parameter	Description
message	Get the field names of this message.
enumeration	Stores a pointer to the enumeration.

### See Also

[tibemsMsgEnum](#)

## tibemsMapMsg\_ItemExists

*Function*

### Purpose

Test if a named pair exists.

### C Declaration

```
tibems_status tibemsMapMsg_ItemExists(
    tibemsMapMsg message,
    const char* name,
    tibems_bool* exists );
```

### COBOL Call

```
CALL "tibemsMapMsg_ItemExists"
    USING BY VALUE message,
         BY REFERENCE name,
         BY REFERENCE exists,
         RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Test in the body of this map message.
name	Test for a pair with this name.
exists	The function stores the boolean result of the test.

## tibemsMapMsg\_Set

*Function*

### Purpose

Set a name-value pair in a map message.

### Single Value C Declarations

```
tibems_status tibemsMapMsg_SetBoolean(
    tibemsMapMsg message,
    const char* name,
    tibems_bool value );
```

```
tibems_status tibemsMapMsg_SetByte(
    tibemsMapMsg message,
    const char* name,
    tibems_byte value );
```

```
tibems_status tibemsMapMsg_SetChar(
    tibemsMapMsg message,
    const char* name,
    tibems_wchar value );
```

```
tibems_status tibemsMapMsg_SetDouble(
    tibemsMapMsg message,
```

```

    const char* name,
    tibems_double value );

tibems_status tibemsMapMsg_SetFloat(
    tibemsMapMsg message,
    const char* name,
    tibems_float value );

tibems_status tibemsMapMsg_SetInt(
    tibemsMapMsg message,
    const char* name,
    tibems_int value );

tibems_status tibemsMapMsg_SetLong(
    tibemsMapMsg message,
    const char* name,
    tibems_long value );

tibems_status tibemsMapMsg_SetShort(
    tibemsMapMsg message,
    const char* name,
    tibems_short value );

tibems_status tibemsMapMsg_SetString(
    tibemsMapMsg message,
    const char* name,
    const char* value );

```

### Array C Declarations

```

tibems_status tibemsMapMsg_SetDoubleArray(
    tibemsMapMsg message,
    const char* name,
    const tibems_double* value,
    tibems_uint count );

tibems_status tibemsMapMsg_SetFloatArray(
    tibemsMapMsg message,
    const char* name,
    const tibems_float* value,
    tibems_uint count );

tibems_status tibemsMapMsg_SetIntArray(
    tibemsMapMsg message,
    const char* name,
    const tibems_int* value,
    tibems_uint count );

tibems_status tibemsMapMsg_SetLongArray(
    tibemsMapMsg message,
    const char* name,
    const tibems_long* value,
    tibems_uint count );

tibems_status tibemsMapMsg_SetShortArray(
    tibemsMapMsg message,
    const char* name,
    const tibems_short* value,
    tibems_uint count );

```

### Nested Message C Declarations

```

tibems_status tibemsMapMsg_SetMapMsg(
    tibemsMapMsg message,
    const char* name,
    tibemsMsg mapMsg,
    tibems_bool takeOwnership );

tibems_status tibemsMapMsg_SetStreamMsg(
    tibemsMsg message,
    const char* name,

```

```
tibemsMsg streamMsg,
tibems_bool takeOwnership);
```

## COBOL Call

```
CALL "tibemsMapMsg_SetBoolean"
    USING BY VALUE message,
          BY REFERENCE name,
          BY VALUE value,
          RETURNING tibems-status
END-CALL.
```

```
CALL "tibemsMapMsg_SetByte"
    USING BY VALUE message,
          BY REFERENCE name,
          BY VALUE value,
          RETURNING tibems-status
END-CALL.
```

```
CALL "tibemsMapMsg_SetChar"
    USING BY VALUE message,
          BY REFERENCE name,
          BY VALUE value,
          RETURNING tibems-status
END-CALL.
```

```
CALL "tibemsMapMsg_SetDouble"
    USING BY VALUE message,
          BY REFERENCE name,
          BY VALUE value,
          RETURNING tibems-status
END-CALL.
```

```
CALL "tibemsMapMsg_SetDoubleArray"
    USING BY VALUE message,
          BY REFERENCE name,
          BY REFERENCE value,
          BY VALUE count,
          RETURNING tibems-status
END-CALL.
```

```
CALL "tibemsMapMsg_SetFloat"
    USING BY VALUE message,
          BY REFERENCE name,
          BY VALUE value,
          RETURNING tibems-status
END-CALL.
```

```
CALL "tibemsMapMsg_SetFloatArray"
    USING BY VALUE message,
          BY REFERENCE name,
          BY REFERENCE value,
          BY VALUE count,
          RETURNING tibems-status
END-CALL.
```

```
CALL "tibemsMapMsg_SetInt"
    USING BY VALUE message,
          BY REFERENCE name,
          BY VALUE value,
          RETURNING tibems-status
END-CALL.
```

```
CALL "tibemsMapMsg_SetIntArray"
    USING BY VALUE message,
          BY REFERENCE name,
          BY REFERENCE value,
          BY VALUE count,
          RETURNING tibems-status
END-CALL.
```

```

CALL "tibemsMapMsg_SetLong"
  USING BY VALUE message,
        BY REFERENCE name,
        BY VALUE value,
        RETURNING tibems-status
END-CALL.

CALL "tibemsMapMsg_SetLongArray"
  USING BY VALUE message,
        BY REFERENCE name,
        BY REFERENCE value,
        BY VALUE count,
        RETURNING tibems-status
END-CALL.

CALL "tibemsMapMsg_SetMapMsg"
  USING BY VALUE message,
        BY REFERENCE name,
        BY VALUE mapMsg,
        BY VALUE tibems-Boolean,
        RETURNING tibems-status
END-CALL.

CALL "tibemsMapMsg_SetShort"
  USING BY VALUE message,
        BY REFERENCE name,
        BY VALUE value,
        RETURNING tibems-status
END-CALL.

CALL "tibemsMapMsg_SetShortArray"
  USING BY VALUE message,
        BY REFERENCE name,
        BY REFERENCE value,
        BY VALUE count,
        RETURNING tibems-status
END-CALL.

CALL "tibemsMapMsg_SetStreamMsg"
  USING BY VALUE message,
        BY REFERENCE name,
        BY VALUE streamMsg,
        BY VALUE tibems-Boolean,
        RETURNING tibems-status
END-CALL.

CALL "tibemsMapMsg_SetString"
  USING BY VALUE message,
        BY REFERENCE name,
        BY REFERENCE value,
        RETURNING tibems-status
END-CALL.

```



message, streamMsg and mapMsg have usage pointer.

### Parameters

Parameter	Description
message	Set the pair in the body of this map message.
name	Set the pair with this name. Field names must not be null, and must not be empty strings.
value	Associate this value with the name.

Parameter	Description
count	Array functions set array values of this size (must be a positive number).
mapMsg	tibemsMapMsg_SetMapMsg sets this map message as a nested value.
takeOwnership	<p>Nested message functions use this parameter to control ownership of nested messages.</p> <p>When this argument is TIBEMS_TRUE, the call increments the reference count of the nested message. This action prevents other calls from destroying the nested message improperly.</p> <p>We recommend that all calls supply TIBEMS_TRUE.</p>

### Remarks

The JMS specification defines functions to set name-value pairs in a MapMessage.

### Extensions

Array functions and nested message functions extend the JMS specification. Use them only with SmartSockets or Rendezvous messages. Programs that use these extensions might be non-compliant, and cannot interoperate with other JMS providers.

### See Also

[tibemsMapMsg\\_Get](#)

[tibemsMapMsg\\_SetBytes](#)

## tibemsMapMsg\_SetBytes

*Function*

### Purpose

Set a byte array as a named value in a map message.

### C Declaration

```
tibems_status tibemsMapMsg_SetBytes(
    tibemsMapMsg message,
    const char* name,
    void* bytes,
    tibems_uint bytesSize );

tibems_status tibemsMapMsg_SetReferencedBytes(
    tibemsMapMsg message,
    const char* name,
    void* bytes,
    tibems_uint bytesSize );
```

### COBOL Call

```
CALL "tibemsMapMsg_SetBytes"
    USING BY VALUE message,
          BY REFERENCE name,
          BY REFERENCE bytes,
          BY VALUE bytesSize,
          RETURNING tibems-status
END-CALL.
```

```
CALL "tibemsMapMsg_SetReferencedBytes"
```

```

    USING BY VALUE message,
          BY REFERENCE name,
          BY REFERENCE bytes,
          BY VALUE bytesSize,
          RETURNING tibems-status
END-CALL.

```



message has usage pointer.

### Parameters

Parameter	Description
message	Set the name and value pair in the body of this map message.
name	Set the pair with this name.
bytes	Associate this byte array value with the name.
bytesSize	Set a byte array value of this length.

### Remarks

`tibemsMapMsg_SetBytes` copies the byte array into the map message field. The program may free the original byte array after this call returns.

`tibemsMapMsg_SetReferencedBytes` adds a reference to the byte array, but does not copy the bytes. When the byte array is very large, it can be more efficient to avoid making a copy. However, the program must not free nor modify the original byte array until after freeing the map message.

## tibemsObjectMsg

*Type*

### Purpose

A message containing a serializable object.

### Related Types

[tibemsMsg](#)

Function	Description
<a href="#">tibemsObjectMsg_Create</a>	Create an object message.
<a href="#">tibemsObjectMsg_GetObjectBytes</a>	Get the byte sequence representing a serialized object from a message.
<a href="#">tibemsObjectMsg_SetObjectBytes</a>	Set the byte sequence of an object message.

### Object Messages in C

Object messages are used for objects created in .NET or Java. A C program can create, receive, and send object messages. For example, a C application can forward or store an object generated from a Java or .NET application.

C programs cannot create object messages. However, a C program can receive an object message from a Java or .NET program, and forward it, or store it for later resending.

### Serialization

The C library neither serializes objects nor reassembles them from bytes.

## tibemsObjectMsg\_Create

*Function*

### Purpose

Create an object message.

### C Declaration

```
tibems_status tibemsObjectMsg_Create(
    tibemsObjectMsg* message );
```

### COBOL Call

```
CALL "tibemsObjectMsg_Create"
    USING BY REFERENCE message,
    RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Store a pointer to the new message.

### Remarks

Create a new object message.

When your application creates a message, it also allocates storage for that message. This storage must subsequently be freed by a call to `tibemsMsg_Destroy`.

### See Also

[tibemsMsg\\_Create](#)

[tibemsMsg\\_Destroy](#)

## tibemsObjectMsg\_GetObjectBytes

*Function*

### Purpose

Get the byte sequence representing a serialized object from a message.

### C Declaration

```
tibems_status tibemsObjectMsg_GetObjectBytes(
    tibemsObjectMsg message,
    void** bytes,
    tibems_uint* byteSize );
```

## COBOL Call

```
CALL "tibemsObjectMsg_GetObjectBytes"
    USING BY VALUE message,
          BY REFERENCE bytes,
          BY REFERENCE byteSize,
          RETURNING tibems-status
END-CALL.
```



message and bytes have usage pointer.

## Parameters

Parameter	Description
message	Get bytes (representing an object) from this message.
bytes	Store a pointer to the byte sequence (within the message).
byteSize	Store the length of the byte sequence.

## Remarks

When the message does not contain an object (because none has been set), this function places null in the bytes argument, and zero in the byteSize argument.

## tibemsObjectMsg\_SetObjectBytes

*Function*

### Purpose

Set the byte sequence of an object message.

### C Declaration

```
tibems_status tibemsObjectMsg_SetObjectBytes(
    tibemsObjectMsg message,
    const void* bytes,
    tibems_uint byteSize );
```

## COBOL Call

```
CALL "tibemsObjectMsg_SetObjectBytes"
    USING BY VALUE message,
          BY REFERENCE bytes,
          BY VALUE byteSize,
          RETURNING tibems-status
END-CALL.
```



message has usage pointer.

## Parameters

Parameter	Description
message	Put bytes (representing an object) into this message.
bytes	Use these bytes (representing a serialized object) as the message body data.

Parameter	Description
<code>byteSize</code>	Length of the byte sequence.

### Remarks

Setting the content of an object message stores a snapshot of the object. Subsequent changes to the original object or its serialized representation (as a byte sequence) do not affect the message.

## tibemsStreamMsg

*Type*

### Purpose

A message containing a stream of data items.

### Related Types

[tibemsMsg](#)

Member	Description
<a href="#">tibemsStreamMsg_Create</a>	Create a stream message.
<a href="#">tibemsStreamMsg_FreeField</a>	Free storage allocated during the reading of a stream message.
<a href="#">tibemsStreamMsg_Read</a>	Read primitive datatypes from a stream message.
<a href="#">tibemsStreamMsg_ReadBytes</a>	Read a byte array from a stream message.
<a href="#">tibemsStreamMsg_ReadField</a>	Read a field from a stream message.
<a href="#">tibemsStreamMsg_Reset</a>	Set the read position to the beginning of the stream, and mark the message body as read-only.
<a href="#">tibemsStreamMsg_Write</a>	Write data to a stream message.
<a href="#">tibemsBytesMsg_WriteBytes</a>	Write bytes from a byte array to a stream message.

### Remarks

Each datum in the stream must be a primitive type, or an object representation of a primitive type.

### Extensions

TIBCO Enterprise Message Service extends the `MapMessage` and `StreamMessage` body types in two ways. These extensions allow TIBCO Enterprise Message Service to exchange messages with TIBCO Rendezvous and ActiveEnterprise formats that have certain features not available within the JMS specification.

- You can insert another `MapMessage` or `StreamMessage` instance as a submessage into a `MapMessage` or `StreamMessage`, generating a series of nested messages, instead of a flat message.
- You can use arrays as well as primitive types for the values.

These extensions add considerable flexibility to the two body types. However, they are extensions and therefore not compliant with JMS specifications. Extended messages are tagged as extensions with the vendor property tag `JMS_TIBCO_MSG_EXT`.

For more information on message compatibility with Rendezvous messages, see *Message Body* in *TIBCO Enterprise Message Service User's Guide*.

## tibemsStreamMsg\_Create

*Function*

### Purpose

Create a stream message.

### C Declaration

```
tibems_status tibemsStreamMsg_Create(
    tibemsStreamMsg* message );
```

### COBOL Call

```
CALL "tibemsStreamMsg_Create"
    USING BY REFERENCE message,
        RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Store a pointer to the new message.

### Remarks

Create a new stream message.

When your application creates a message, it also allocates storage for that message. This storage must subsequently be freed by a call to `tibemsMsg_Destroy`.

### See Also

[tibemsMsg\\_Create](#)

[tibemsMsg\\_Destroy](#)

## tibemsStreamMsg\_FreeField

*Function*

### Purpose

Free storage allocated during the reading of a stream message.

## C Declaration

```
void tibemsStreamMsg_FreeField(
    tibemsMsgField* field );
```

## COBOL Call

```
CALL "tibemsStreamMsg_FreeField"
    USING BY REFERENCE field
END-CALL.
```

## Parameters

Parameter	Description
field	Free the storage associated with this field struct.

## Remarks

Each successful call to `tibemsStreamMsg_ReadField` allocates storage for a `tibemsMsgField`. Programs that call `tibemsStreamMsg_ReadField` must subsequently call this function to free that allocated storage.

This function does not need to be called after `tibemsMapMsg_GetField`.

## See Also

[tibemsStreamMsg\\_ReadField](#)

[tibemsMsgField](#)

## tibemsStreamMsg\_Read

*Function*

### Purpose

Read primitive datatypes from a stream message.

### C Declaration

```
tibems_status tibemsStreamMsg_ReadBoolean(
    tibemsStreamMsg message,
    tibems_bool* value );

tibems_status tibemsStreamMsg_ReadByte(
    tibemsStreamMsg message,
    tibems_byte* value );

tibems_status tibemsStreamMsg_ReadChar(
    tibemsStreamMsg message,
    tibems_wchar* value );

tibems_status tibemsStreamMsg_ReadDouble(
    tibemsStreamMsg message,
    tibems_double* value );

tibems_status tibemsStreamMsg_ReadFloat(
    tibemsStreamMsg message,
    tibems_float* value );

tibems_status tibemsStreamMsg_ReadInt(
    tibemsStreamMsg message,
    tibems_int* value );
```

```

tibems_status tibemsStreamMsg_ReadLong(
    tibemsStreamMsg message,
    tibems_long* value );

tibems_status tibemsStreamMsg_ReadShort(
    tibemsStreamMsg message,
    tibems_short* value );

tibems_status tibemsStreamMsg_ReadString(
    tibemsStreamMsg message,
    char** value );

```

## COBOL Call

```

CALL "tibemsStreamMsg_ReadBoolean"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_ReadByte"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_ReadChar"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_ReadDouble"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_ReadFloat"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_ReadInt"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_ReadLong"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_ReadShort"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_ReadString"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.

```



message has usage pointer.

value has usage pointer only in `tibemsStreamMsg_ReadString` (but not in the other calls documented in this group).

### Parameters

Parameter	Description
message	Read a field struct from this message.
value	Store a pointer to the field struct.

### Remarks

Each call reads a unit of data from the stream, and advances the read position so that the next read call gets the next datum. (Other read functions are documented on separate pages.)

### See Also

[tibemsStreamMsg\\_ReadBytes](#)

[tibemsStreamMsg\\_ReadField](#)

## tibemsStreamMsg\_ReadBytes

*Function*

### Purpose

Read a byte array from a stream message.

### C Declaration

```
tibems_status tibemsStreamMsg_ReadBytes(
    tibemsStreamMsg message,
    void** value,
    tibems_uint* length );
```

### COBOL Call

```
CALL "tibemsStreamMsg_ReadBytes"
    USING BY VALUE message,
          BY REFERENCE value,
          BY REFERENCE size,
          RETURNING tibems-status
END-CALL.
```



message and value have usage pointer.

### Parameters

Parameter	Description
message	Read a byte array from this message.
value	Store a pointer to the byte sequence (within the message).
length	Store the actual number of bytes read.

## Remarks

Each call reads bytes from the stream into the byte array, and advances the read position so that the next read call gets the next datum. (Other read functions are documented on separate pages.)

This call uses the `length` parameter to return the actual number of bytes read. When the call cannot read even one byte, the `length` is -1.

A program that calls this function must call it repeatedly until it returns -1, indicating that the program has extracted the complete set of bytes. Only then may the program call another read function.

## See Also

[tibemsStreamMsg\\_Read](#)

[tibemsStreamMsg\\_ReadField](#)

## tibemsStreamMsg\_ReadField

*Function*

### Purpose

Read a field from a stream message.

### C Declaration

```
tibems_status tibemsStreamMsg_ReadField(
    tibemsStreamMsg message,
    tibemsMsgField* value );
```

### COBOL Call

```
CALL "tibemsStreamMsg_ReadField"
    USING BY VALUE message,
         BY REFERENCE value,
         RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Read a field struct from this message.
value	The function stores a pointer to the field struct.

### Remarks

Each call reads a field from the stream, and advances the read position so that the next read call gets the next datum. (Other read functions are documented on separate pages.)

### Freeing Fields

Each successful call to `tibemsStreamMsg_ReadField` creates a field struct. Field structs can contain data in allocated storage. Programs that call `tibemsStreamMsg_ReadField` *must* subsequently call [tibemsStreamMsg\\_FreeField](#) to free that allocated storage.

**See Also**

[tibemsStreamMsg\\_FreeField](#)

**tibemsStreamMsg\_Reset**

*Function*

**Purpose**

Set the read position to the beginning of the stream, and mark the message body as read-only.

**C Declaration**

```
tibems_status tibemsStreamMsg_Reset(
    tibemsStreamMsg message );
```

**COBOL Call**

```
CALL "tibemsStreamMsg_Reset"
    USING BY VALUE message,
        RETURNING tibems-status
END-CALL.
```



message has usage pointer.

**Parameters**

Parameter	Description
message	Reset the read position of this message.

**Remarks**

This call prepares a message body for reading, as if the message were newly received. Contrast [tibemsMsg\\_ClearBody](#), which clears a message body in preparation for writing, as if it were newly created.

**tibemsStreamMsg\_Write**

*Function*

**Purpose**

Write data to a stream message.

**Single Value C Declarations**

```
tibems_status tibemsStreamMsg_WriteBoolean(
    tibemsStreamMsg message,
    tibems_bool value );

tibems_status tibemsStreamMsg_WriteByte(
    tibemsStreamMsg message,
    tibems_byte value );

tibems_status tibemsStreamMsg_WriteChar(
    tibemsStreamMsg message,
    tibems_wchar value );

tibems_status tibemsStreamMsg_WriteDouble(
    tibemsStreamMsg message,
    tibems_double value );
```

```

tibems_status tibemsStreamMsg_WriteFloat(
    tibemsStreamMsg message,
    tibems_float value );

tibems_status tibemsStreamMsg_WriteInt(
    tibemsStreamMsg message,
    tibems_int value );

tibems_status tibemsStreamMsg_WriteLong(
    tibemsStreamMsg message,
    tibems_long value );

tibems_status tibemsStreamMsg_WriteShort(
    tibemsStreamMsg message,
    tibems_short value );

tibems_status tibemsStreamMsg_WriteString(
    tibemsStreamMsg message,
    char* value );

```

### Array C Declarations

```

tibems_status tibemsStreamMsg_WriteDoubleArray(
    tibemsMsg message,
    const tibems_double* value,
    tibems_int count );

tibems_status tibemsStreamMsg_WriteFloatArray(
    tibemsMsg message,
    const tibems_float* value,
    tibems_int count );

tibems_status tibemsStreamMsg_WriteIntArray(
    tibemsMsg message,
    const tibems_int* value,
    tibems_int count );

tibems_status tibemsStreamMsg_WriteLongArray(
    tibemsMsg message,
    const tibems_long* value,
    tibems_int count );

tibems_status tibemsStreamMsg_WriteShortArray(
    tibemsMsg message,
    const tibems_short* value,
    tibems_int count );

```

### Nested Message C Declarations

```

tibems_status tibemsStreamMsg_WriteMapMsg(
    tibemsMsg message,
    tibemsMsg value );

tibems_status tibemsStreamMsg_WriteStreamMsg(
    tibemsMsg message,
    tibemsMsg value );

```



Nested messages are an extension of the JMS specification. Programs that use this feature are non-compliant.

### COBOL Call

```

CALL "tibemsStreamMsg_WriteBoolean"
    USING BY VALUE message,
         BY VALUE value,
         RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_WriteByte"

```

```

        USING BY VALUE message,
              BY VALUE value,
              RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_WriteChar"
    USING BY VALUE message,
          BY VALUE value,
          RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_WriteDouble"
    USING BY VALUE message,
          BY VALUE value,
          RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_WriteDoubleArray"
    USING BY VALUE message,
          BY REFERENCE value,
          BY VALUE count,
          RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_WriteFloat"
    USING BY VALUE message,
          BY VALUE value,
          RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_WriteFloatArray"
    USING BY VALUE message,
          BY REFERENCE value,
          BY VALUE count,
          RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_WriteInt"
    USING BY VALUE message,
          BY VALUE value,
          RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_WriteIntArray"
    USING BY VALUE message,
          BY REFERENCE value,
          BY VALUE count,
          RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_WriteLong"
    USING BY VALUE message,
          BY VALUE value,
          RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_WriteLongArray"
    USING BY VALUE message,
          BY REFERENCE value,
          BY VALUE count,
          RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_WriteMapMsg"
    USING BY VALUE message,
          BY VALUE value,
          RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_WriteShort"
    USING BY VALUE message,
          BY VALUE value,

```

```

        RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_WriteShortArray"
    USING BY VALUE message,
          BY REFERENCE value,
          BY VALUE count,
          RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_WriteStreamMsg"
    USING BY VALUE message,
          BY VALUE value,
          RETURNING tibems-status
END-CALL.

CALL "tibemsStreamMsg_WriteString"
    USING BY VALUE message,
          BY REFERENCE value,
          RETURNING tibems-status
END-CALL.

```



message has usage pointer.

value has usage pointer only in `tibemsStreamMsg_WriteStreamMsg` and `tibemsStreamMsg_WriteMapMsg` (but not in the other calls documented in this group).

### Parameters

Parameter	Description
message	Write a datum to this stream message.
value	Write this datum. Arrays must not be null.
count	Array functions set array values of this size.

### Remarks

Each call writes a data value to the stream, and advances the write position so that the next write call appends to the new end of the stream.

### See Also

[tibemsStreamMsg\\_WriteBytes](#)

## tibemsStreamMsg\_WriteBytes

*Function*

### Purpose

Write bytes from a byte array to a stream message.

### C Declaration

```

tibems_status tibemsStreamMsg_WriteBytes(
    tibemsStreamMsg message,
    void* value,
    tibems_uint length );

```

## COBOL Call

```
CALL "tibemsStreamMsg_WriteBytes"
    USING BY VALUE message,
          BY REFERENCE value,
          BY VALUE length,
          RETURNING tibems-status
END-CALL.
```



message has usage pointer.

### Parameters

Parameter	Description
message	Write to the byte stream of this message.
value	Copy bytes from this byte array to the message.
length	Write this number of bytes from the byte array to the message.

### Remarks

Each call writes bytes from the byte array into the stream, and advances the write position.

## tibemsTextMsg

*Type*

### Purpose

A message containing a text string.

### Related Types

[tibemsMsg](#)

### Remarks

Messages with this body type contain a single value, which is a string.

Function	Description
<a href="#">tibemsTextMsg_Create</a>	Create a text message.
<a href="#">tibemsTextMsg_GetText</a>	Get the string data from a text message.
<a href="#">tibemsTextMsg_SetText</a>	Set the data string of a text message.

## tibemsTextMsg\_Create

*Function*

### Purpose

Create a text message.

## C Declaration

```
tibems_status tibemsTextMsg_Create(
    tibemsTextMsg* message );
```

## COBOL Call

```
CALL "tibemsTextMsg_Create"
    USING BY REFERENCE message,
        RETURNING tibems-status
END-CALL.
```



message has usage pointer.

## Parameters

Parameter	Description
message	Store a pointer to the new message.

## Remarks

Create a new text message.

When your application creates a message, it also allocates storage for that message. This storage must subsequently be freed by a call to `tibemsMsg_Destroy`.

## See Also

[tibemsMsg\\_Create](#)

[tibemsMsg\\_Destroy](#)

[tibemsSession\\_CreateTextMessage](#)

## tibemsTextMsg\_GetText

*Function*

## Purpose

Get the string data from a text message.

## C Declaration

```
tibems_status tibemsTextMsg_GetText(
    tibemsTextMsg message,
    const char** text );
```

## COBOL Call

```
CALL "tibemsTextMsg_GetText"
    USING BY VALUE message,
        BY REFERENCE text,
        RETURNING tibems-status
END-CALL.
```



message and text have usage pointer.

## Parameters

Parameter	Description
message	Get the string from this message.
text	Store a pointer to the string (within the message).

## Remarks

When the message does not contain any text (because none has been set), this function sets its text parameter to null.

## tibemsTextMsg\_SetText

*Function*

### Purpose

Set the data string of a text message.

### C Declaration

```
tibems_status tibemsTextMsg_SetText(
    tibemsTextMsg message,
    const char* text );
```

### COBOL Call

```
CALL "tibemsTextMsg_SetText"
    USING BY VALUE message,
         BY REFERENCE text,
         RETURNING tibems-status
END-CALL.
```



message has usage pointer.

## Parameters

Parameter	Description
message	Put the text into this message.
text	Copy this string into the message body.

## tibemsData

*Type*

### Purpose

Union type that covers all possible datatypes in `atibemsMsgField` struct.

### C Declaration

```
typedef union {
    tibems_bool boolValue;
    tibems_byte byteValue;
    tibems_short shortValue;
```

```

tibems_wchar wcharValue;
tibems_int intValue;
tibems_long longValue;
tibems_float floatValue;
tibems_double doubleValue;
char* utf8Value;
void* bytesValue;
struct __tibemsMsg* msgValue;
void* arrayValue;
} tibemsData;

```

## COBOL

```

05 MsgFld-data.
  10 MFD                PIC  X(8).
  10 MFD-boolValue      redefines MFD          PIC  S9(8) BINARY.
  10 MFD-byteValue      redefines MFD          PIC  X(1) USAGE DISPLAY.
  10 MFD-shortValue     redefines MFD          PIC  S9(4) BINARY.
  10 MFD-wcharValue     redefines MFD          PIC  9(4) COMPUTATIONAL-5.
  10 MFD-intValue       redefines MFD          PIC  S9(9) BINARY.
  10 MFD-longValue      redefines MFD          PIC  S9(18) COMPUTATIONAL-5.
  10 MFD-floatValue     redefines MFD          USAGE COMPUTATIONAL-1.
  10 MFD-doubleValue    redefines MFD          USAGE COMPUTATIONAL-2.
  10 MFD-utf8Value      redefines MFD          USAGE POINTER.
  10 MFD-bytesValue     redefines MFD          USAGE POINTER.
  10 MFD-msgValue       redefines MFD          USAGE POINTER.
  10 MFD-arrayValue     redefines MFD          USAGE POINTER.

```

## See Also

[tibemsMsgField](#)

## tibemsDeliveryMode

*Type*

### Purpose

Define delivery mode constants.

### C Declaration

```

typedef enum {
    TIBEMS_NON_PERSISTENT,
    TIBEMS_PERSISTENT,
    TIBEMS_RELIABLE
} tibemsDeliveryMode;

```

## COBOL

```

01 TIBEMS-DELIVERY-MODES.
  05 tibemsDeliveryMode      PIC  S9(8) BINARY.
     88 TIBEMS-NON-PERSISTENT VALUE  1.
     88 TIBEMS-PERSISTENT   VALUE  2.
     88 TIBEMS-RELIABLE     VALUE 22.

```

Member	Description
TIBEMS_NON_PERSISTENT	Non-persistent delivery.
TIBEMS_PERSISTENT	Persistent delivery.
TIBEMS_RELIABLE	Reliable delivery mode is a TIBCO proprietary extension that offers increased performance of the message producers. See also <a href="#">RELIABLE_DELIVERY</a> in <i>TIBCO Enterprise Message Service User's Guide</i> .

## tibemsNpCheckMode

*Type*

### Purpose

Define when a producer should check the result of sending a NON\_PERSISTENT message.

### C Declaration

```
typedef enum
{
    NPSEND_CHECK_DEFAULT      = 0,
    NPSEND_CHECK_ALWAYS      = 1,
    NPSEND_CHECK_NEVER       = 2,
    NPSEND_CHECK_TEMP_DEST   = 3,
    NPSEND_CHECK_AUTH        = 4,
    NPSEND_CHECK_TEMP_AUTH   = 5
} tibemsNpCheckMode;
```

### COBOL

```
01 TIBEMS-NPCHECK-MODES .
   05 tibemsNpCheckMode      PIC S9(8) BINARY .
   88 NPSEND_CHECK_DEFAULT   VALUE 0 .
   88 NPSEND_CHECK_ALWAYS    VALUE 1 .
   88 NPSEND_CHECK_NEVER     VALUE 2 .
   88 NPSEND_CHECK_TEMP_DEST VALUE 3 .
   88 NPSEND_CHECK_AUTH      VALUE 4 .
   88 NPSEND_CHECK_TEMP_AUTH VALUE 5 .
```

Member	Description
NPSEND_CHECK_DEFAULT	Define default check mode for sending a NON_PERSISTENT message.
NPSEND_CHECK_ALWAYS	Define mode when producer always checks result of sending a NON_PERSISTENT message.
NPSEND_CHECK_NEVER	Define mode when producer never checks result of sending a NON_PERSISTENT message.
NPSEND_CHECK_TEMP_DEST	Define mode when producer checks result of sending a NON_PERSISTENT message only when sending into temporary destination.

Member	Description
NPSEND_CHECK_AUTH	Define mode when producer checks result of sending a NON_PERSISTENT message only when server authorization is enabled.
NPSEND_CHECK_TEMP_AUTH	Define mode when producer checks result of sending a NON_PERSISTENT message when sending into temporary destination or if server authorization is enabled.

## tibemsMsgCompletionCallback

*Type*

### Purpose

Asynchronously process a message after sending. This is equivalent to a completion listener in the Java and .NET API.

### C Declaration

```
typedef void (*tibemsMsgCompletionCallback) (
    tibemsMsg msg,
    tibems_status status,
    void* closure );
```

### Parameters

Parameter	Description
msg	Receive the message object.
status	Receive the status of the associated send call.
closure	Receive the closure argument, which your program registered in the call to the asynchronous send.

### Remarks

To asynchronously send messages, your program can define callback functions of this type, and register them when sending asynchronously (using [tibemsMsgProducer\\_AsyncSend](#) or a related function).

If the send of the message was successful, the status parameter is TIBEMS\_OK. If the send is not successful, the status is an error code indicating the type of failure that occurred.

This call is not supported in COBOL.

### See Also

[tibemsMsgProducer](#)

[Sending Asynchronously](#) in the *TIBCO Enterprise Message Service User's Guide*.

## tibemsMsgEnum

*Type*

### Purpose

Enumerate the properties of a message, or the field names of a map message.

### Remarks

[tibemsMsg\\_GetPropertyNames](#) and [tibemsMapMsg\\_GetMapNames](#) create instances of this type.

Function	Description
<a href="#">tibemsMsgEnum_Destroy</a>	Destroy a message enumerator.
<a href="#">tibemsMsgEnum_GetNextName</a>	Get the next item from a message enumerator.

### See Also

[tibemsMsg\\_GetPropertyNames](#)

[tibemsMapMsg\\_GetMapNames](#)

## tibemsMsgEnum\_Destroy

*Function*

### Purpose

Destroy a message enumerator.

### C Declaration

```
tibems_status tibemsMsgEnum_Destroy(  
    tibemsMsgEnum enumeration );
```

### COBOL Call

```
CALL "tibemsMsgEnum_Destroy"  
    USING BY VALUE enumeration,  
         RETURNING tibems-status  
END-CALL.
```



enumeration has usage pointer.

### Parameters

Parameter	Description
enumeration	Destroy this enumerator.

## tibemsMsgEnum\_GetNextName

*Function*

### Purpose

Get the next item from a message enumerator.

## C Declaration

```
tibems_status tibemsMsgEnum_GetNextName(
    tibemsMsgEnum enumeration,
    const char** name );
```

## COBOL Call

```
CALL "tibemsMsgEnum_GetNextName"
    USING BY VALUE enumeration,
         BY REFERENCE name,
         RETURNING tibems-status
END-CALL.
```



enumeration and name have usage pointer.

## Parameters

Parameter	Description
enumeration	Get the next item from this enumerator.
name	Store the next item.

## tibemsMsgField

*Type*

### Purpose

Represents a message field or property.

## C Declaration

```
typedef struct {
    tibems_byte type;
    tibems_int size;
    tibems_int count;
    tibemsData data;
} tibemsMsgField
```

## COBOL

```
01 tibemsMsgField.
   05 MsgFld-type          PIC X(1).
      88 TIBEMS-NULL      VALUE X'00'.
      88 TIBEMS-BOOL      VALUE X'01'.
      88 TIBEMS-BYTE      VALUE X'02'.
      88 TIBEMS-WCHAR     VALUE X'03'.
      88 TIBEMS-SHORT     VALUE X'04'.
      88 TIBEMS-INT       VALUE X'05'.
      88 TIBEMS-LONG      VALUE X'06'.
      88 TIBEMS-FLOAT     VALUE X'07'.
      88 TIBEMS-DOUBLE    VALUE X'08'.
      88 TIBEMS-UTF8      VALUE X'09'.
      88 TIBEMS-BYTES     VALUE X'0A'.
      88 TIBEMS-MAP-MSG   VALUE X'0B'.
      88 TIBEMS-STREAM-MSG VALUE X'0C'.
      88 TIBEMS-SHORT-ARRAY VALUE X'14'.
      88 TIBEMS-INT-ARRAY VALUE X'15'.
      88 TIBEMS-LONG-ARRAY VALUE X'16'.
      88 TIBEMS-FLOAT-ARRAY VALUE X'17'.
      88 TIBEMS-DOUBLE-ARRAY VALUE X'18'.
   05 Filler              PIC X(3).
```

```

05 MsgFld-size          PIC S9(9) BINARY.
05 MsgFld-count        PIC S9(9) BINARY.
05 Filler              PIC X(4).
05 MsgFld-data.
10 MFD                  PIC X(8).

```

### Remarks

Any message can have property values. Only map messages and stream messages can have fields.

Field	Description
type	One-byte indicator of the field's datatype; for values, see the following table.
size	Size of the data (in bytes). Zero is a special value, indicating that the size is unknown.
count	Number of elements in the array.
data	Actual data in the field, or property value.

### Message Field Type Indicators

Constant	Value	Comment
TIBEMS_NULL	0	
TIBEMS_BOOL	1	
TIBEMS_BYTE	2	
TIBEMS_WCHAR	3	wide character; 2 bytes
TIBEMS_SHORT	4	
TIBEMS_INT	5	
TIBEMS_LONG	6	
TIBEMS_FLOAT	7	
TIBEMS_DOUBLE	8	
TIBEMS_UTF8	9	UTF8-encoded string
TIBEMS_BYTES	10	
TIBEMS_MAP_MSG	11	
TIBEMS_STREAM_MSG	12	
TIBEMS_SHORT_ARRAY	20	
TIBEMS_INT_ARRAY	21	

Constant	Value	Comment
TIBEMS_LONG_ARRAY	22	
TIBEMS_FLOAT_ARRAY	23	
TIBEMS_DOUBLE_ARRAY	24	

Function	Description
<code>tibemsMsgField_Print</code>	Print a message field.
<code>tibemsMsgField_PrintToBuffer</code>	Print a message field into a buffer.

### See Also

`tibemsMapMsg_GetField`, listed at [tibemsMapMsg\\_Get](#)

[tibemsMsg\\_Get](#) Property

[tibemsStreamMsg\\_FreeField](#)

[tibemsStreamMsg\\_ReadField](#)

[tibemsMsgEnum\\_GetNextName](#)

## tibemsMsgField\_Print

*Function*

### Purpose

Print a message field.

### C Declaration

```
void tibemsMsgField_Print(
    tibemsMsgField* field );
```

### COBOL Call

```
CALL "tibemsMsgField_Print"
    USING BY REFERENCE field,
END-CALL.
```

### Parameters

Parameter	Description
<code>field</code>	Print this message field instance.

### Remarks

`tibemsMsgField_Print` prints to stdout in the format *FieldDataType:Value*.

## tibemsMsgField\_PrintToBuffer

*Function*

### Purpose

Print a message field into a buffer.

### C Declaration

```
tibems_status tibemsMsgField_PrintToBuffer(
    tibemsMsgField* field,
    char* buffer,
    tibems_int maxlen);
```

### COBOL Call

```
CALL "tibemsMsgField_PrintToBuffer"
    USING BY REFERENCE field,
    BY REFERENCE buffer,
    BY VALUE maxlen,
    RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
field	Print this message field instance.
buffer	Location to store the string representation of the message.
maxlen	Size of the buffer.

### Remarks

tibemsMsgField\_PrintToBuffer prints to a buffer, in the format *FieldDataType:Value*.

Status Code	Description
TIBEMS_INSUFFICIENT_BUFFER	Buffer is too small to hold the data.

## tibemsMsgType

*Type*

### Purpose

Enumerated constants representing message body types.

### C Declaration

```
typedef enum {
    TIBEMS_MESSAGE_UNKNOWN,
    TIBEMS_MESSAGE,
    TIBEMS_BYTES_MESSAGE,
    TIBEMS_MAP_MESSAGE,
    TIBEMS_OBJECT_MESSAGE,
    TIBEMS_STREAM_MESSAGE,
    TIBEMS_TEXT_MESSAGE,
```

```
TIBEMS_MESSAGE_UNDEFINED
} tibemsMsgType;
```

## COBOL

```
01 TIBEMS-MSG-TYPES.
   05 tibemsMsgType          PIC S9(8) BINARY.
      88 TIBEMS-MESSAGE-UNKNOWN VALUE 0.
      88 TIBEMS-MESSAGE        VALUE 1.
      88 TIBEMS-BYTES-MESSAGE  VALUE 2.
      88 TIBEMS-MAP-MESSAGE    VALUE 3.
      88 TIBEMS-OBJECT-MESSAGE VALUE 4.
      88 TIBEMS-STREAM-MESSAGE VALUE 5.
      88 TIBEMS-TEXT-MESSAGE   VALUE 6.
      88 TIBEMS-MESSAGE-UNDEFINED VALUE 256.
```

## See Also

[tibemsMsg\\_GetBodyType](#)

# Destination

This section documents the functions that create and modify destination objects. The destination of a message determines where the message is sent.

## Destination Overview

Destination objects represent destinations within the EMS server—the queues and topics to which programs send messages, and from which they receive messages.

Queues deliver each message to exactly one consumer. Topics deliver each message to every subscriber. Queues and topics can be static, dynamic or temporary.

Aspect	Static	Dynamic	Temporary
Purpose	Static destinations let administrators configure EMS behavior at the enterprise level. Administrators define these administered objects, and client programs use them—relieving program developers and end users of the responsibility for correct configuration.	Dynamic destinations give client programs the flexibility to define destinations as needed for short-term use.	Temporary destinations are ideal for limited-scope uses, such as reply subjects.
Scope of Delivery	Static destinations support concurrent use. That is, several client processes (and in several threads within a process) can create local objects denoting the destination, and consume messages from it.	Dynamic destinations support concurrent use. That is, several client processes (and in several threads within a process) can create local objects denoting the destination, and consume messages from it.	Temporary destinations support only local use. That is, only the client connection that created a temporary destination can consume messages from it. However, servers connected by routes do exchange messages sent to temporary topics.
Creation	Administrators create static destinations using EMS server administration tools or API.	If the server configuration permits dynamic destinations, client programs can create one in two steps: <ol style="list-style-type: none"> <li>1. Create a local destination object; see <a href="#">tibemsSession</a>.</li> <li>2. Send a message to that destination, or create a consumer for it. Either of these actions automatically creates the destination in the server.</li> </ol>	Client programs create temporary destinations; see <a href="#">tibemsSession</a> .

Aspect	Static	Dynamic	Temporary
Lookup	Client programs lookup static destinations by name. Successful lookup returns a local object representation of the destination; see <a href="#">tibemsLookupContext_Lookup</a> .	Not applicable.	Not applicable.
Duration	A static destination remains in the server until an administrator explicitly deletes it.	<p>A dynamic destination remains in the server as long as at least one client actively uses it. The server automatically deletes it (at a convenient time) when all applicable conditions are true:</p> <ul style="list-style-type: none"> <li>• <b>Topic or Queue</b> all client programs that access the destination have disconnected</li> <li>• <b>Topic</b> no offline durable subscribers exist for the topic</li> <li>• <b>Queue</b> no messages are stored in the queue</li> </ul>	A temporary destination remains in the server either until the client that created it explicitly deletes it, or until the client disconnects from the server.

## tibemsDestinationType

*Type*

### Purpose

Enumerated constants representing destination types.

### C Declaration

```
typedef enum {
    TIBEMS_UNKNOWN,
    TIBEMS_QUEUE,
    TIBEMS_TOPIC
} tibemsDestinationType;
```

### COBOL

```
01 TIBEMS-DESTINATION-TYPES.
   05 tibemsDestinationType    PIC S9(8) BINARY.
      88 TIBEMS-UNKNOWN        VALUE 0.
      88 TIBEMS-QUEUE          VALUE 1.
      88 TIBEMS-TOPIC          VALUE 2.
      88 TIBEMS-UNDEFINED      VALUE 256.
```

### See Also

[tibemsDestination\\_Create](#)

[tibemsDestination\\_GetType](#)

## tibemsDestination

*Type*

### Purpose

Represent a named queue or topic in the server.

### Remarks

Administrators define destinations in the server. Client programs access them using functions of [tibemsLookupContext](#).

Function	Description
<a href="#">tibemsDestination_Copy</a>	Create an independent copy of a destination object.
<a href="#">tibemsDestination_Create</a>	Create a destination object.
<a href="#">tibemsDestination_Destroy</a>	Destroy a destination object.
<a href="#">tibemsDestination_GetName</a>	Get the name of a destination object.
<a href="#">tibemsDestination_GetType</a>	Get the type of a destination object.

### Related Types

[tibemsQueue](#)

[tibemsTemporaryQueue](#)

[tibemsTopic](#)

[tibemsTemporaryTopic](#)

### See Also

[tibemsMsg\\_GetDestination](#)

[tibemsMsg\\_SetDestination](#)

[tibemsLookupContext](#)

## tibemsDestination\_Copy

*Function*

### Purpose

Create an independent copy of a destination object.

### C Declaration

```
tibems_status tibemsDestination_Copy(
    tibemsDestination destination,
    tibemsDestination* copy );
```

## COBOL Call

```
CALL "tibemsDestination_Copy"
    USING BY VALUE destination,
          BY REFERENCE copy,
          RETURNING tibems-status
END-CALL.
```



destination has usage pointer.

### Parameters

Parameter	Description
destination	Copy this existing destination object.
copy	Store the new copy.

## tibemsDestination\_Create

*Function*

### Purpose

Create a destination object.

### C Declaration

```
tibems_status tibemsDestination_Create(
    tibemsDestination* destination,
    tibemsDestinationType type,
    const char* name );
```

## COBOL Call

```
CALL "tibemsDestination_Create"
    USING BY REFERENCE destination,
          BY VALUE type,
          BY REFERENCE name,
          RETURNING tibems-status
END-CALL.
```



destination has usage pointer.

### Parameters

Parameter	Description
destination	Store the new destination in this location.
type	Create a destination of this type (queue or topic).
name	Create a destination with this name.

### See Also

[tibemsDestinationType](#)

## tibemsDestination\_Destroy

*Function*

### Purpose

Destroy a destination object.

### C Declaration

```
tibems_status tibemsDestination_Destroy(
    tibemsDestination destination );
```

### COBOL Call

```
CALL "tibemsDestination_Destroy"
    USING BY VALUE destination,
         RETURNING tibems-status
END-CALL.
```



destination has usage pointer.

### Parameters

Parameter	Description
destination	Destroy this destination.

### Remarks

Note that a destination object retrieved from a message, consumer, or producer object should only be destroyed by calling the destroy function of the corresponding message, consumer, or producer object.

## tibemsDestination\_GetName

*Function*

### Purpose

Get the name of a destination object.

### C Declaration

```
tibems_status tibemsDestination_GetName(
    tibemsDestination destination,
    char* name,
    tibems_int name_len );
```

### COBOL Call

```
CALL "tibemsDestination_GetName"
    USING BY VALUE destination,
         BY REFERENCE name,
         BY VALUE name-len
    RETURNING tibems-status
END-CALL.
```



destination has usage pointer.

## Parameters

Parameter	Description
destination	Get the name of this destination.
name	The function copies the name to this location.
name_len	Length of the name buffer.

## Remarks

A null character terminates the copied name string.

Your program must allocate the name buffer, and pass its length to the function. If the length of the name is greater than the size of the buffer provided, the entire destination name may not be copied. The buffer size is determined by the `TIBEMS_DESTINATION_MAX` constant. Constants such as `TIBEMS_DESTINATION_MAX` are located in the `tibems/types.h` header file.

## tibemsDestination\_GetType

*Function*

### Purpose

Get the type of a destination object.

### C Declaration

```
tibems_status tibemsDestination_GetType(
    tibemsDestination destination,
    tibemsDestinationType* type );
```

### COBOL Call

```
CALL "tibemsDestination_GetType"
    USING BY VALUE destination,
         BY REFERENCE type,
         RETURNING tibems-status
END-CALL.
```



destination has usage pointer.

## Parameters

Parameter	Description
destination	Get the type of this destination.
type	The function stores the type in this location.

## See Also

[tibemsDestinationType](#)

## tibemsQueue

*Type*

### Purpose

Queues deliver each message to exactly one consumer.

Function	Description
<a href="#">tibemsQueue_Create</a>	Create a queue object.
<a href="#">tibemsQueue_Destroy</a>	Destroy a queue object.
<a href="#">tibemsQueue_GetQueueName</a>	Get the name of a queue object.

### Related Types

[tibemsDestination](#)

[tibemsTemporaryQueue](#)

## tibemsQueue\_Create

*Function*

### Purpose

Create a queue object.

### C Declaration

```
tibems_status tibemsQueue_Create(
    tibemsQueue* queue,
    const char* queueName );
```

### COBOL Call

```
CALL "tibemsQueue_Create"
    USING BY REFERENCE queue,
         BY REFERENCE queueName,
         RETURNING tibems-status
END-CALL.
```



queue has usage pointer.

### Parameters

Parameter	Description
queue	Store the queue object in this location.
queueName	Create a local queue instance with this name.

**Remarks**

This call creates only local objects (within the program). It does not attempt to bind the local queue object to the corresponding server object until the program creates a [tibemsMsgConsumer](#) or a [tibemsMsgProducer](#) that uses the queue.

The bind can fail for the following reasons:

- Authorization is enabled for the queue and the user does not have the appropriate permissions.
- The queue does not exist and the server cannot dynamically create the queue.

**See Also**

[tibemsLookupContext](#)

**tibemsQueue\_Destroy**

*Function*

**Purpose**

Destroy a queue object.

**C Declaration**

```
tibems_status tibemsQueue_Destroy(
    tibemsQueue queue );
```

**COBOL Call**

```
CALL "tibemsQueue_Destroy"
    USING BY VALUE queue,
         RETURNING tibems-status
END-CALL.
```



queue has usage pointer.

**Parameters**

Parameter	Description
queue	Destroy this queue.

**tibemsQueue\_GetQueueName**

*Function*

**Purpose**

Get the name of a queue object.

**C Declaration**

```
tibems_status tibemsQueue_GetQueueName(
    tibemsQueue queue,
    char* name,
    tibems_int name_len );
```

## COBOL Call

```
CALL "tibemsQueue_GetQueueName"
    USING BY VALUE queue,
          BY REFERENCE name,
          BY VALUE name-len
          RETURNING tibems-status
END-CALL.
```



queue has usage pointer.

### Parameters

Parameter	Description
queue	Get the name of this queue.
name	The function copies the name to this location.
name_len	Length of the name buffer.

### Remarks

A null character terminates the copied name string.

Your program must allocate the name buffer, and pass its length to the function. If the length of the queue name is greater than the size of the buffer provided, the entire queue name may not be copied. The buffer size is determined by the `TIBEMS_DESTINATION_MAX` constant. Constants such as `TIBEMS_DESTINATION_MAX` are located in the `tibems/types.h` header file.

## tibemsTemporaryQueue

*Type*

### Purpose

Programs can use temporary queues as reply destinations.

### Remarks

Programs create temporary queues using [tibemsSession\\_CreateTemporaryQueue](#).

A temporary queue exists only for the duration of the session's connection, and is available only within that connection.

Only consumers associated with the same connection as the temporary queue can consume messages from it.

All functions that accept a queue or a generic destination as an argument also accept a temporary queue.

Function	Description
<a href="#">tibemsSession_CreateTemporaryQueue</a>	Create a temporary queue.
<a href="#">tibemsSession_DeleteTemporaryQueue</a>	Delete a temporary queue.

**Related Types**[tibemsDestination](#)[tibemsQueue](#)**tibemsTemporaryTopic***Type***Purpose**

Programs can use temporary topics as reply destinations.

**Remarks**

Programs create temporary topics using [tibemsSession\\_CreateTemporaryTopic](#).

A temporary topic exists only for the duration of the session's connection, and is available only within that connection.

Only consumers associated with the same connection as the temporary topic can consume messages from it.

Servers connected by routes do exchange messages sent to temporary topics.

All functions that accept a topic or a generic destination as an argument also accept a temporary queue.

Function	Description
<a href="#">tibemsSession_CreateTemporaryTopic</a>	Create a temporary topic.
<a href="#">tibemsSession_DeleteTemporaryTopic</a>	Delete a temporary topic.

**Related Types**[tibemsDestination](#)[tibemsTopic](#)**tibemsTopic***Type***Purpose**

Topics deliver each message to multiple consumers.

Function	Description
<a href="#">tibemsTopic_Create</a>	Create a topic object.
<a href="#">tibemsTopic_Destroy</a>	Destroy a topic object.
<a href="#">tibemsTopic_GetTopicName</a>	Get the name of a topic object.

**Related Types**[tibemsDestination](#)

[tibemsTemporaryTopic](#)

## tibemsTopic\_Create

*Function*

### Purpose

Create a topic object.

### C Declaration

```
tibems_status tibemsTopic_Create(
    tibemsTopic* topic,
    const char* topicName );
```

### COBOL Call

```
CALL "tibemsTopic_Create"
    USING BY REFERENCE topic,
         BY REFERENCE topicName,
         RETURNING tibems-status
END-CALL.
```



topic has usage pointer.

### Parameters

Parameter	Description
topic	Store the topic object in this location.
topicName	Create a local topic instance with this name.

### Remarks

This constructor creates only local objects (within the program). It does not attempt to bind the local topic object to the corresponding server object until the program creates a [tibemsMsgConsumer](#) or a [tibemsMsgProducer](#) that uses the topic.

The bind can fail for the following reasons:

- Authorization is enabled for the topic and the user does not have the appropriate permissions.
- The topic does not exist and the server cannot dynamically create the topic.

### See Also

[tibemsLookupContext](#)

## tibemsTopic\_Destroy

*Function*

### Purpose

Destroy a topic object.

### C Declaration

```
tibems_status tibemsTopic_Destroy(
    tibemsTopic topic );
```

**COBOL Call**

```
CALL "tibemsTopic_Destroy"
    USING BY VALUE topic,
          RETURNING tibems-status
END-CALL.
```



topic has usage pointer.

**Parameters**

Parameter	Description
topic	Destroy this topic object.

**tibemsTopic\_GetTopicName**

*Function*

**Purpose**

Get the name of a topic object.

**C Declaration**

```
tibems_status tibemsTopic_GetTopicName(
    tibemsTopic topic,
    char* name,
    tibems_int name_len );
```

**COBOL Call**

```
CALL "tibemsTopic_GetTopicName"
    USING BY VALUE topic,
          BY REFERENCE name,
          BY VALUE name-len,
          RETURNING tibems-status
END-CALL.
```



topic has usage pointer.

**Parameters**

Parameter	Description
topic	Get the name of this topic.
name	The function copies the name to this location.
name_len	Length of the name buffer.

**Remarks**

A null character terminates the copied name string.

Your program must allocate the name buffer, and pass its length to the function. If the length of the topic name is greater than the size of the buffer provided, the entire topic name may not be copied. The buffer size is determined by the TIBEMS\_DESTINATION\_MAX constant. Constants such as TIBEMS\_DESTINATION\_MAX are located in the tibems/types.h header file.

# Consumer

This section documents the functions that create and modify message consumers. A message consumer receives messages from a destination.

## tibemsMsgConsumer

*Type*

### Purpose

Consume messages from a destination.

### Remarks

Consumers can receive messages synchronously (using the receive functions), or asynchronously.

Consumers can receive messages asynchronously using callback functions.

Clients create message consumers using functions of a [tibemsSession](#) object.

Function	Description
<a href="#">tibemsMsgConsumer_Close</a>	Close a message consumer and releases associated storage.
<a href="#">tibemsMsgConsumer_GetDestination</a>	Get the destination from a message consumer.
<a href="#">tibemsMsgConsumer_GetMsgListener</a>	Get the message callback and closure data from a consumer.
<a href="#">tibemsMsgConsumer_GetMsgSelector</a>	Get the message selector from a consumer.
<a href="#">tibemsMsgConsumer_GetNoLocal</a>	Get the no local property of a message consumer.
<a href="#">tibemsMsgConsumer_Receive</a>	Receive a message (synchronous).
<a href="#">tibemsMsgConsumer_ReceiveNoWait</a>	Receive a message (synchronous, non-blocking).
<a href="#">tibemsMsgConsumer_ReceiveTimeout</a>	Receive a message (synchronous, blocks up to a time limit).
<a href="#">tibemsMsgConsumer_SetMsgListener</a>	Set the message callback and closure data of a consumer.

### See Also

[tibemsMsgCallback](#)

[tibemsSession\\_CreateConsumer](#)

## tibemsMsgConsumer\_Close

*Function*

### Purpose

Close a message consumer and release associated storage.

## C Declaration

```
tibems_status tibemsMsgConsumer_Close(
    tibemsMsgConsumer msgConsumer );
```

## COBOL Call

```
CALL "tibemsMsgConsumer_Close"
    USING BY VALUE msgConsumer,
         RETURNING tibems-status
END-CALL.
```



msgConsumer has usage pointer.

## Parameters

Parameter	Description
msgConsumer	Close this consumer.

## Remarks

If a receive call or a message listener callback is in progress, then this function waits until that call returns, and then closes the consumer.

This call also notifies the server that the client program is closing the consumer. In response, the server stops sending message data to the consumer.

Your program must explicitly close all consumers that it creates.

## See Also

[tibemsMsgConsumer\\_Receive](#)

[tibemsSession\\_CreateConsumer](#)

## tibemsMsgConsumer\_GetDestination

*Function*

### Purpose

Get the destination from a message consumer.

## C Declaration

```
tibems_status tibemsMsgConsumer_GetDestination(
    tibemsMsgConsumer msgConsumer,
    tibemsDestination* destination );
```

## COBOL Call

```
CALL "tibemsMsgConsumer_GetDestination"
    USING BY VALUE msgConsumer,
         BY REFERENCE destination,
         RETURNING tibems-status
END-CALL.
```



msgConsumer and destination have usage pointer.

## Parameters

Parameter	Description
msgConsumer	Get the destination from this consumer.
destination	Store the destination in this location.

## Remarks

The consumer consumes messages from this destination.

Programs set this destination when creating the consumer, and cannot subsequently change it.

## tibemsMsgConsumer\_GetMsgListener

*Function*

### Purpose

Get the message callback and closure data from a consumer.

### C Declaration

```
tibems_status tibemsMsgConsumer_GetMsgListener(
    tibemsMsgConsumer msgConsumer,
    tibemsMsgCallback* callbackPtr,
    void** closure );
```

## Parameters

Parameter	Description
msgConsumer	Get the listener from this consumer.
callbackPtr	Store a pointer to the callback in this location.
closure	Store a pointer to the closure data in this location. This pointer is passed into the message listener callback.

## Remarks

EMS C programs can implement a message listener as a callback function paired with closure data. This call extracts these items from a consumer object.

Your program implements the callback, and registers it by calling [tibemsMsgConsumer\\_SetMsgListener](#). When a message arrives, the consumer calls the callback.

This call is not supported in COBOL.

## See Also

[tibemsMsgConsumer\\_SetMsgListener](#)

[tibemsMsgCallback](#)

## tibemsMsgConsumer\_GetMsgSelector

*Function*

### Purpose

Get the message selector from a consumer.

### C Declaration

```
tibems_status tibemsMsgConsumer_GetMsgSelector(
    tibemsMsgConsumer msgConsumer,
    const char** selectorPtr );
```

### COBOL Call

```
CALL "tibemsMsgConsumer_GetMsgSelector"
    USING BY VALUE msgConsumer,
         BY REFERENCE selectorPtr,
         RETURNING tibems-status
END-CALL.
```



msgConsumer and selectorPtr have usage pointer.

### Parameters

Parameter	Description
msgConsumer	Get the listener from this consumer.
selectorPtr	Store a pointer to the selector string in this location.

### Remarks

A message selector restricts the set of messages that the consumer receives to those that match the selector; see [Message Selectors](#).

Programs can set this property only when creating the consumer object; see [tibemsSession\\_CreateConsumer](#).

### See Also

[tibemsMsgCallback](#)

## tibemsMsgConsumer\_GetNoLocal

*Function*

### Purpose

Get the no local property of a message consumer.

### C Declaration

```
tibems_status tibemsMsgConsumer_GetNoLocal(
    tibemsMsgConsumer msgConsumer,
    tibems_bool* noLocal );
```

### COBOL Call

```
CALL "tibemsMsgConsumer_GetNoLocal"
    USING BY VALUE msgConsumer,
```

```

        BY REFERENCE noLocal,
        RETURNING tibems-status
END-CALL.

```



msgConsumer has usage pointer.

### Parameters

Parameter	Description
msgConsumer	Get the property from this consumer.
noLocal	Store the property value in this location.

### Remarks

When true, the consumer does not receive messages sent through the same server connection (that is, the connection associated with the consumer).

Programs set this property when creating the consumer, and cannot subsequently change it.

This property is only associated with topics, and does not apply to queues.

## tibemsMsgConsumer\_Receive

*Function*

### Purpose

Receive a message (synchronous).

### C Declaration

```

tibems_status tibemsMsgConsumer_Receive(
    tibemsMsgConsumer msgConsumer,
    tibemsMsg* message );

```

### COBOL Call

```

CALL "tibemsMsgConsumer_Receive"
    USING BY VALUE msgConsumer,
         BY REFERENCE message,
         RETURNING tibems-status
END-CALL.

```



msgConsumer and message have usage pointer.

### Parameters

Parameter	Description
msgConsumer	Receive a message through this consumer.
message	Store a pointer to the inbound message in this location.

### Remarks

This function consumes the next message from the consumer's destination.

When the destination does not have any messages ready, this function blocks:

- If a message arrives at the destination, this call immediately consumes that message and returns.
- If another thread closes the consumer, this call returns `TIBEMS_INTR`.

When calling `tibemsMsgConsumer_Receive` from a transaction, the consumer retains the message until transaction commits.

### See Also

[tibemsMsgConsumer\\_ReceiveNoWait](#)

[tibemsMsgConsumer\\_ReceiveTimeout](#)

## tibemsMsgConsumer\_ReceiveNoWait

*Function*

### Purpose

Receive a message (synchronous, non-blocking).

### C Declaration

```
tibems_status tibemsMsgConsumer_ReceiveNoWait(
    tibemsMsgConsumer msgConsumer,
    tibemsMsg* message);
```

### COBOL Call

```
CALL "tibemsMsgConsumer_ReceiveNoWait"
    USING BY VALUE msgConsumer,
         BY REFERENCE message,
         RETURNING tibems-status
END-CALL.
```



`msgConsumer` and `message` have usage pointer.

### Parameters

Parameter	Description
<code>msgConsumer</code>	Receive a message through this consumer.
<code>msg</code>	The function stores a pointer to the inbound message in this location.

### Remarks

When the destination has at least one message ready, this function immediately returns the next message.

When the destination does *not* have any messages ready, this function immediately returns `TIBEMS_NOT_FOUND`.

When calling `receive` within a transaction, the consumer retains the message until transaction commits.

Note that this function should not be used if the destination property `prefetch=none`.

### See Also

[tibemsMsgConsumer\\_Receive](#)

[tibemsMsgConsumer\\_ReceiveTimeout](#)

## tibemsMsgConsumer\_ReceiveTimeout

*Function*

### Purpose

Receive a message (synchronous, blocks up to a time limit).

### C Declaration

```
tibems_status tibemsMsgConsumer_ReceiveTimeout(
    tibemsMsgConsumer msgConsumer,
    tibemsMsg* message,
    tibems_long timeout );
```

### COBOL Call

```
CALL "tibemsMsgConsumer_ReceiveTimeout"
    USING BY VALUE msgConsumer,
         BY REFERENCE message,
         BY VALUE timeout,
         RETURNING tibems-status
END-CALL.
```



msgConsumer and message have usage pointer.

### Parameters

Parameter	Description
msgConsumer	Receive a message through this consumer.
msg	Store a pointer to the inbound message in this location.
timeout	Wait no longer than this interval (in milliseconds) for a message to arrive. Zero is a special value, which specifies no timeout (block indefinitely).

### Remarks

This function consumes the next message from the consumer's destination. When the destination does not have any messages ready, this function blocks:

- If a message arrives at the destination, this call immediately consumes that message and returns.
- If the (non-zero) timeout elapses before a message arrives, this call returns [TIBEMS\\_TIMEOUT](#).
- If another thread closes the consumer, this call returns [TIBEMS\\_INTR](#).

When calling receive within a transaction, the consumer retains the message until transaction commits.

### See Also

[tibemsMsgConsumer\\_Receive](#)

[tibemsMsgConsumer\\_ReceiveNoWait](#)

## tibemsMsgConsumer\_SetMsgListener

*Function*

### Purpose

Set the message callback and closure data of a consumer.

### C Declaration

```
tibems_status tibemsMsgConsumer_SetMsgListener(
    tibemsMsgConsumer msgConsumer,
    tibemsMsgCallback callback,
    void* closure );
```

### Parameters

Parameter	Description
msgConsumer	Set the listener of this consumer.
callback	Use this callback function.
closure	Use this closure data.

### Remarks

EMS C programs can implement a message listener as a callback function paired with closure data. This call sets these items in a consumer object.

Your program implements the callback, and registers it by calling this function. When a message arrives, the consumer calls the callback.

This call is not supported in COBOL.

### See Also

[tibemsMsgCallback](#)

## tibemsMsgCallback

*Type*

### Purpose

Asynchronously process an arriving message.

### C Declaration

```
typedef void (*tibemsMsgCallback) (
    tibemsMsgConsumer msgConsumer,
    tibemsMsg msg,
    void* closure );
```

### Parameters

Parameter	Description
msgConsumer	Receive the consumer object.

Parameter	Description
<code>msg</code>	Receive the message object.
<code>closure</code>	Receive the closure argument, which your program registered on the consumer.

### Remarks

To asynchronously receive messages, your program can define callback functions of this type, and register them with a consumer (using [tibemsMsgConsumer\\_SetMsgListener](#) or a related function). When a message arrives, the consumer calls its callback.

This call is not supported in COBOL.

### Serialization

In compliance with the JMS specification, sessions distribute messages to consumers in serial (non-concurrent) fashion.

### See Also

[tibemsMsgConsumer](#)

# Producer

Message producers send messages to destinations on the server.

## tibemsMsgProducer

*Type*

### Purpose

Send message to destinations on the server.

### Remarks

Clients use message producers to send messages. A message producer object can store several parameters that affect the messages it sends.

Clients create message producers using functions of a `tibemsSession` object.

Function	Description
<a href="#">tibemsMsgProducer_AsyncSend</a> <a href="#">tibemsMsgProducer_AsyncSendEx</a> <a href="#">tibemsMsgProducer_AsyncSendToDestination</a> <a href="#">tibemsMsgProducer_AsyncSendToDestinationEx</a>	Asynchronously send a message.
<a href="#">tibemsMsgProducer_Close</a>	Destroy the producer object; reclaim resources.
<a href="#">tibemsMsgProducer_GetDeliveryDelay</a>	Get the delivery delay property of a producer object.
<a href="#">tibemsMsgProducer_GetDeliveryMode</a>	Get the delivery mode property of a producer object.
<a href="#">tibemsMsgProducer_GetDestination</a>	Get the destination of a message producer object.
<a href="#">tibemsMsgProducer_GetDisableMessageID</a>	Get the disable message ID property of a producer object.
<a href="#">tibemsMsgProducer_GetDisableMessageTimestamp</a>	Get the disable message timestamp property of a producer object.
<a href="#">tibemsMsgProducer_GetNPSTransmitCheckMode</a>	Get the mode that defines when a producer checks the result of sending a NON_PERSISTENT message.
<a href="#">tibemsMsgProducer_GetPriority</a>	Get the priority property of a producer object.

Function	Description
<a href="#">tibemsMsgProducer_GetTimeToLive</a>	Get the time-to-live property of a producer object.
<a href="#">tibemsMsgProducer_Send</a> <a href="#">tibemsMsgProducer_SendEx</a> <a href="#">tibemsMsgProducer_SendToDestination</a> <a href="#">tibemsMsgProducer_SendToDestinationEx</a>	Send a message.
<a href="#">tibemsMsgProducer_SetDeliveryDelay</a>	Set the delivery delay property of a producer object.
<a href="#">tibemsMsgProducer_SetDeliveryMode</a>	Set the delivery mode property of a producer object.
<a href="#">tibemsMsgProducer_SetDisableMessageID</a>	<a href="#">Set the disable message ID property of a producer object.</a>
<a href="#">tibemsMsgProducer_SetDisableMessageTimestamp</a>	Set the disable message timestamp property of a producer object.
<a href="#">tibemsMsgProducer_SetNPSendCheckMode</a>	Set when a producer should check the result of sending a NON_PERSISTENT message.
<a href="#">tibemsMsgProducer_SetPriority</a>	Set the priority property of a producer object.
<a href="#">tibemsMsgProducer_SetTimeToLive</a>	Set the time-to-live property of a producer object.

**See Also**

[tibemsSession\\_CreateProducer](#)

**tibemsMsgProducer\_AsyncSend**

*Function*

**Purpose**

Asynchronously send a message.

**C Declaration**

```

tibems_status tibemsMsgProducer_AsyncSend(
    tibemsMsgProducer msgProducer,
    tibemsMsg message
    tibemsMsgCompletionCallback asyncSendCallback,
    void* asyncSendClosure );

tibems_status tibemsMsgProducer_AsyncSendEx(
    tibemsMsgProducer msgProducer,
    tibemsMsg message,
    tibems_int deliveryMode,

```

```

tibems_int priority,
tibems_long timeToLive
tibemsMsgCompletionCallback asyncSendCallback,
void* asyncSendClosure );

tibems_status tibemsMsgProducer_AsyncSendToDestination(
tibemsMsgProducer msgProducer,
tibemsDestination destination,
tibemsMsg message
tibemsMsgCompletionCallback asyncSendCallback,
void* asyncSendClosure );

tibems_status tibemsMsgProducer_AsyncSendToDestinationEx(
tibemsMsgProducer msgProducer,
tibemsDestination destination,
tibemsMsg message,
tibemsDeliveryMode deliveryMode,
tibems_int priority,
tibems_long timeToLive
tibemsMsgCompletionCallback asyncSendCallback,
void* asyncSendClosure );

```

### Parameters

Parameter	Description
msgProducer	Send a message through this producer object.
destination	Send the message to this destination (queue or topic).  Other send calls send the message to the producer's default destination. When the producer does not specify a default, the send call must supply this parameter.
message	Send this message object.
deliveryMode	Send the message with this delivery mode.  This argument is an enumerated value (see <a href="#">tibemsDeliveryMode</a> ).  Other send calls send the message with the producer's default delivery mode.
priority	Send the message with this priority.  Priority affects the order in which the server delivers messages to consumers (higher values first). The JMS specification defines ten levels of priority value, from zero (lowest priority) to 9 (highest priority). The specification suggests that clients consider 0–4 as gradations of normal priority, and priorities 5–9 as gradations of expedited priority.  Other send calls send the message with the producer's default priority.

Parameter	Description
timeToLive	<p>Use this value (in milliseconds) to compute the message expiration.</p> <ul style="list-style-type: none"> <li>If the time-to-live is non-zero, the expiration is the sum of that time-to-live and the sending client's current time (GMT). This rule applies even within sessions with transaction semantics—the timer begins with the send call, not the commit call.</li> <li>If the time-to-live is zero, then expiration is also zero—indicating that the message never expires.</li> </ul> <p>Other send calls use the producer's default value to compute expiration.</p> <p>Whenever your application uses non-zero values for message expiration or time-to-live, you must ensure that clocks are synchronized among all the host computers that send and receive messages. Synchronize clocks to a tolerance that is a very small fraction of the smallest or time-to-live.</p>
asyncSendCallback	A callback to be invoked when a message has successfully been sent.
asyncSendCallbackClosure	Data to be passed into the asyncSendCallback.

### Remarks

These calls are not supported in COBOL.

## tibemsMsgProducer\_Close

*Function*

### Purpose

Destroy the producer object; reclaim resources.

### C Declaration

```
tibems_status tibemsMsgProducer_Close(
    tibemsMsgProducer msgProducer );
```

### COBOL Call

```
CALL "tibemsMsgProducer_Close"
    USING BY VALUE msgProducer,
    RETURNING tibems-status
END-CALL.
```



msgProducer has usage pointer.

**Parameters**

Parameter	Description
msgProducer	Close this producer.

**Remarks**

This call also notifies the server that the client program is closing the producer. In response, the server reclaims storage associated with the producer.

Your program must explicitly close all producers that it creates.

**See Also**

[tibemsSession\\_CreateProducer](#)

**tibemsMsgProducer\_GetDeliveryDelay**

*Function*

**Purpose**

Get the delivery delay property of a producer object.

**C Declaration**

```
tibems_status tibemsMsgProducer_GetDeliveryDelay(
    tibemsMsgProducer msgProducer,
    tibems_long* deliveryDelay );
```

**COBOL Call**

```
CALL "tibemsMsgProducer_GetDeliveryDelay"
    USING BY VALUE msgProducer,
         BY REFERENCE deliveryDelay,
         RETURNING tibems-status
END-CALL.
```



msgProducer has usage pointer.

**Parameters**

Parameter	Description
msgProducer	Get the property from this producer.
deliveryDelay	Store the property in this location.

**Remarks**

Get the minimum length of time, in milliseconds, that must elapse after a message is sent before the JMS provider may deliver the message to a consumer.

**See Also**

[tibemsMsg\\_GetDeliveryTime](#)

[tibemsMsgProducer\\_SetDeliveryDelay](#)

## tibemsMsgProducer\_GetDeliveryMode

*Function*

### Purpose

Get the delivery mode property of a producer object.

### C Declaration

```
tibems_status tibemsMsgProducer_GetDeliveryMode(
    tibemsMsgProducer msgProducer,
    tibems_int* deliveryMode );
```

### COBOL Call

```
CALL "tibemsMsgProducer_GetDeliveryMode"
    USING BY VALUE msgProducer,
         BY REFERENCE deliveryMode,
         RETURNING tibems-status
END-CALL.
```



msgProducer has usage pointer.

### Parameters

Parameter	Description
msgProducer	Get the property from this producer.
deliveryMode	Store the property in this location.

### Remarks

Delivery mode instructs the server concerning persistent storage.

Programs can use this property to define a default delivery mode for messages that this producer sends. Individual sending calls can override this default value.

For values, see the type [tibemsDeliveryMode](#).

## tibemsMsgProducer\_GetDestination

*Function*

### Purpose

Get the destination of a message producer object.

### C Declaration

```
tibems_status tibemsMsgProducer_GetDestination(
    tibemsMsgProducer msgProducer,
    tibemsDestination* destination );
```

### COBOL Call

```
CALL "tibemsMsgProducer_GetDestination"
    USING BY VALUE msgProducer,
         BY REFERENCE destination,
         RETURNING tibems-status
END-CALL.
```



`msgProducer` and `destination` have usage pointer.

### Parameters

Parameter	Description
<code>msgProducer</code>	Get the destination from this producer.
<code>destination</code>	Store the destination in this location.

### Remarks

Each send call directs a message to a destination.

Programs can use this property to define a default destination for messages that this producer sends. Individual sending calls can override this default value.

Programs set this destination when creating the sender, and cannot subsequently change it.

## `tibemsMsgProducer_GetDisableMessageID`

*Function*

### Purpose

Get the disable message ID property of a producer object.

### C Declaration

```
tibems_status tibemsMsgProducer_GetDisableMessageID(
    tibemsMsgProducer msgProducer,
    tibems_bool* disable );
```

### COBOL Call

```
CALL "tibemsMsgProducer_GetDisableMessageID"
    USING BY VALUE msgProducer,
         BY REFERENCE disable,
         RETURNING tibems-status
END-CALL.
```



`msgProducer` has usage pointer.

### Parameters

Parameter	Description
<code>msgProducer</code>	Get the property from this producer.
<code>disable</code>	Store the property in this location.

### Remarks

Applications that do not require message IDs can reduce overhead costs by disabling IDs (set this property to true).

## tibemsMsgProducer\_GetDisableMessageTimestamp

*Function*

### Purpose

Get the disable message timestamp property of a producer object.

### C Declaration

```
tibems_status tibemsMsgProducer_GetDisableMessageTimestamp(
    tibemsMsgProducer msgProducer,
    tibems_bool* disable );
```

### COBOL Call

```
CALL "tibemsMsgProducer_GetDisableMessageTimestamp"
    USING BY VALUE msgProducer,
         BY REFERENCE disable,
         RETURNING tibems-status
END-CALL.
```



msgProducer has usage pointer.

### Parameters

Parameter	Description
msgProducer	Get the property from this producer.
disableMessageTimeStamp	Store the property in this location.

### Remarks

Applications that do not require timestamps can reduce overhead costs by disabling timestamps (set this property to true).

## tibemsMsgProducer\_GetNPSTransmitMode

*Function*

### Purpose

Get the mode that defines when a producer checks the result of sending a NON\_PERSISTENT message.

### C Declaration

```
extern tibems_status tibemsMsgProducer_GetNPSTransmitMode(
    tibemsMsgProducer msgProducer,
    tibemsNpCheckMode* mode);
```

### COBOL Call

```
CALL "tibemsMsgProducer_GetNPSTransmitMode"
    USING BY VALUE msgProducer,
         BY REFERENCE mode,
         RETURNING tibems-status
END-CALL.
```



msgProducer has usage pointer.

## Parameters

Parameter	Description
msgProducer	Get the property from this producer.
mode	Store the mode in this location. Must not be null. See <a href="#">tibemsNpCheckMode</a> for details.

## Remarks

This function returns the send check mode set by the [tibemsMsgProducer\\_SetNPSEndCheckMode](#) function. If the returned value is NPSEND\_CHECK\_DEFAULT, then the effective mode can be set by the server parameter applied to all producers.

## See Also

[tibemsMsgProducer\\_SetNPSEndCheckMode](#)

## tibemsMsgProducer\_GetPriority

*Function*

### Purpose

Get the priority property of a producer object.

### C Declaration

```
tibems_status tibemsMsgProducer_GetPriority(
    tibemsMsgProducer msgProducer,
    tibems_int* priority );
```

### COBOL Call

```
CALL "tibemsMsgProducer_GetPriority"
    USING BY VALUE msgProducer,
         BY REFERENCE priority,
         RETURNING tibems-status
END-CALL.
```



msgProducer has usage pointer.

## Parameters

Parameter	Description
msgProducer	Get the property from this producer.
priority	Store the property in this location.

## Remarks

Priority affects the order in which the server delivers messages to consumers (higher values first).

The JMS specification defines ten levels of priority value, from zero (lowest priority) to 9 (highest priority). The specification suggests that clients consider 0–4 as gradations of normal priority, and priorities 5–9 as gradations of expedited priority.

Programs can use this property to define a default priority for messages that this producer sends. Individual sending calls can override this default value.

## tibemsMsgProducer\_GetTimeToLive

*Function*

### Purpose

Get the time-to-live property of a producer object.

### C Declaration

```
tibems_status tibemsMsgProducer_GetTimeToLive(
    tibemsMsgProducer msgProducer,
    tibems_long* timeToLive );
```

### COBOL Call

```
CALL "tibemsMsgProducer_GetTimeToLive"
    USING BY VALUE msgProducer,
         BY REFERENCE timeToLive,
         RETURNING tibems-status
END-CALL.
```



msgProducer has usage pointer.

### Parameters

Parameter	Description
msgProducer	Get the property from this producer.
timeToLive	Store the property in this location.

### Remarks

Time-to-live (in milliseconds) determines the expiration time of a message.

- If the time-to-live is non-zero, the expiration is the sum of that time-to-live and the sending client's current time (GMT). This rule applies even within sessions with transaction semantics—the timer begins with the send call, not the commit call.
- If the time-to-live is zero, then expiration is also zero—indicating that the message never expires.

Programs can use this property to define a default time-to-live for messages that this producer sends. Individual sending calls can override this default value.

Whenever your application uses non-zero values for message expiration or time-to-live, you must ensure that clocks are synchronized among all the host computers that send and receive messages. Synchronize clocks to a tolerance that is a very small fraction of the smallest or time-to-live.

## tibemsMsgProducer\_Send

*Function*

### Purpose

Send a message.

## C Declaration

```

tibems_status tibemsMsgProducer_Send(
    tibemsMsgProducer msgProducer,
    tibemsMsg message );

tibems_status tibemsMsgProducer_SendEx(
    tibemsMsgProducer msgProducer,
    tibemsMsg message,
    tibems_int deliveryMode,
    tibems_int priority,
    tibems_long timeToLive );

tibems_status tibemsMsgProducer_SendToDestination(
    tibemsMsgProducer msgProducer,
    tibemsDestination destination,
    tibemsMsg message );

tibems_status tibemsMsgProducer_SendToDestinationEx(
    tibemsMsgProducer msgProducer,
    tibemsDestination destination,
    tibemsMsg message,
    tibemsDeliveryMode deliveryMode,
    tibems_int priority,
    tibems_long timeToLive );

```

## COBOL Call

```

CALL "tibemsMsgProducer_Send"
    USING BY VALUE msgProducer,
          BY VALUE message,
          RETURNING tibems-status
END-CALL.

CALL "tibemsMsgProducer_SendEx"
    USING BY VALUE msgProducer,
          BY VALUE message,
          BY VALUE deliveryMode,
          BY VALUE priority,
          BY VALUE timeToLive,
          RETURNING tibems-status
END-CALL.

CALL "tibemsMsgProducer_SendToDestination"
    USING BY VALUE msgProducer,
          BY VALUE destination,
          BY VALUE message,
          RETURNING tibems-status
END-CALL.

CALL "tibemsMsgProducer_SendToDestinationEx"
    USING BY VALUE msgProducer,
          BY VALUE destination,
          BY VALUE message,
          BY VALUE deliveryMode,
          BY VALUE priority,
          BY VALUE timeToLive,
          RETURNING tibems-status
END-CALL.

```



msgProducer, message and destination have usage pointer.

## Parameters

Parameter	Description
<code>msgProducer</code>	Send a message through this producer object.
<code>destination</code>	Send the message to this destination (queue or topic).  Other send calls send the message to the producer's default destination. When the producer does not specify a default, the send call must supply this parameter.
<code>message</code>	Send this message object.
<code>deliveryMode</code>	Send the message with this delivery mode.  This argument is an enumerated value (see <code>tibemsDeliveryMode</code> ).  Other send calls send the message with the producer's default delivery mode.
<code>priority</code>	Send the message with this priority.  Priority affects the order in which the server delivers messages to consumers (higher values first). The JMS specification defines ten levels of priority value, from zero (lowest priority) to 9 (highest priority). The specification suggests that clients consider 0–4 as gradations of normal priority, and priorities 5–9 as gradations of expedited priority.  Other send calls send the message with the producer's default priority.
<code>timeToLive</code>	Use this value (in milliseconds) to compute the message expiration.  <ul style="list-style-type: none"> <li>• If the time-to-live is non-zero, the expiration is the sum of that time-to-live and the sending client's current time (GMT). This rule applies even within sessions with transaction semantics—the timer begins with the send call, not the commit call.</li> <li>• If the time-to-live is zero, then expiration is also zero—indicating that the message never expires.</li> </ul> Other send calls use the producer's default value to compute expiration.  Whenever your application uses non-zero values for message expiration or time-to-live, you must ensure that clocks are synchronized among all the host computers that send and receive messages. Synchronize clocks to a tolerance that is a very small fraction of the smallest or time-to-live.

## `tibemsMsgProducer_SetDeliveryDelay`

*Function*

### Purpose

Set the delivery delay property of a producer object.

### C Declaration

```
tibems_status tibemsMsgProducer_SetDeliveryDelay(
    tibemsMsgProducer msgProducer,
    tibems_long deliveryDelay );
```

## COBOL Call

```
CALL "tibemsMsgProducer_SetDeliveryDelay"
    USING BY VALUE msgProducer,
          BY VALUE deliveryDelay,
          RETURNING tibems-status
END-CALL.
```



msgProducer has usage pointer.

## Parameters

Parameter	Description
msgProducer	Set the property of this producer.
deliveryDelay	Set the property to this value.

## Remarks

Set the minimum length of time, in milliseconds, that must elapse after a message is sent before the JMS provider may deliver the message to a consumer.

For transacted sends, this time starts when the client sends the message, not when the transaction is committed.

The default value is zero, indicating no delay in delivery.

## See Also

[tibemsMsg\\_GetDeliveryTime](#)

[tibemsMsgProducer\\_GetDeliveryDelay](#)

## tibemsMsgProducer\_SetDeliveryMode

*Function*

### Purpose

Set the delivery mode property of a producer object.

### C Declaration

```
tibems_status tibemsMsgProducer_SetDeliveryMode(
    tibemsMsgProducer msgProducer,
    tibems_int deliveryMode );
```

## COBOL Call

```
CALL "tibemsMsgProducer_SetDeliveryMode"
    USING BY VALUE msgProducer,
          BY VALUE deliveryMode,
          RETURNING tibems-status
END-CALL.
```



msgProducer has usage pointer.

## Parameters

Parameter	Description
msgProducer	Set the property of this producer.
deliveryMode	Set the property to this value.

## Remarks

Delivery mode instructs the server concerning persistent storage.

Programs can use this property to define a default delivery mode for messages that this producer sends. Individual sending calls can override this default value.

For values, see the type [tibemsDeliveryMode](#).

## tibemsMsgProducer\_SetDisableMessageID

*Function*

### Purpose

Set the disable message ID property of a producer object.

### C Declaration

```
tibems_status tibemsMsgProducer_SetDisableMessageID(
    tibemsMsgProducer msgProducer,
    tibems_bool disable );
```

### COBOL Call

```
CALL "tibemsMsgProducer_SetDisableMessageID"
    USING BY VALUE msgProducer,
         BY VALUE disable,
         RETURNING tibems-status
END-CALL.
```



msgProducer has usage pointer.

## Parameters

Parameter	Description
msgProducer	Set the property of this producer.
disable	Set the property to this value.

## Remarks

Applications that do not require message IDs can reduce overhead costs by disabling IDs (set this property to true).

## tibemsMsgProducer\_SetDisableMessageTimestamp

*Function*

### Purpose

Set the disable message timestamp property of a producer object.

### C Declaration

```
tibems_status tibemsMsgProducer_SetDisableMessageTimestamp(
    tibemsMsgProducer msgProducer,
    tibems_bool disable );
```

### COBOL Call

```
CALL "tibemsMsgProducer_SetDisableMessageTimestamp"
    USING BY VALUE msgProducer,
         BY VALUE disable,
         RETURNING tibems-status
END-CALL.
```



msgProducer has usage pointer.

### Parameters

Parameter	Description
msgProducer	Set the property of this producer.
disable	Set the property to this value.

### Remarks

Applications that do not require timestamps can reduce overhead costs by disabling timestamps (set this property to true).

## tibemsMsgProducer\_SetNPSTransmitCheckMode

*Function*

### Purpose

Set when a producer should check the result of sending a NON\_PERSISTENT message.

### C Declaration

```
extern tibems_status tibemsMsgProducer_SetNPSTransmitCheckMode(
    tibemsMsgProducer msgProducer,
    tibemsNpCheckMode mode);
```

### COBOL Call

```
CALL "tibemsMsgProducer_SetNPSTransmitCheckMode"
    USING BY VALUE msgProducer,
         BY VALUE mode,
         RETURNING tibems-status
END-CALL.
```



msgProducer has usage pointer.

## Parameters

Parameter	Description
msgProducer	Set the property of this producer.
mode	The send check mode. See <a href="#">tibemsNpCheckMode</a> for details.

## Remarks

[tibemsNpCheckMode](#) only applies to messages sent using the NON\_PERSISTENT delivery mode, using non-transactional Session. It does not apply to cases when message was sent using PERSISTENT or RELIABLE\_DELIVERY delivery modes, or if the corresponding Session is transactional.

If the producer's send check mode is not set, it may execute the mode applied globally to all producers via EMS server parameter. Setting any mode other than NPSEND\_CHECK\_DEFAULT unconditionally overrides global setting defined by the server.

Normally applications use the server's setting or configure producers with a specific send mode only once. However, if required, applications may choose to change this mode before sending every message.

If a producer does not check the result of sending a message, it will not know if a problem has occurred and the message was not processed by the server. If the producer checks the result of the send, the send method will receive the server's response to the send and throw the appropriate exception if any problem has occurred. However, checking the result of sending a message will reduce the performance of the producer.

## See Also

[tibemsMsgProducer\\_GetNPSendCheckMode](#)

## tibemsMsgProducer\_SetPriority

*Function*

### Purpose

Set the priority property of a producer object.

### C Declaration

```
tibems_status tibemsMsgProducer_SetPriority(
    tibemsMsgProducer msgProducer,
    tibems_int priority );
```

### COBOL Call

```
CALL "tibemsMsgProducer_SetPriority"
    USING BY VALUE msgProducer,
          BY VALUE priority,
          RETURNING tibems-status
END-CALL.
```



msgProducer has usage pointer.

## Parameters

Parameter	Description
msgProducer	Set the property of this producer.
priority	Set the property to this value.

## Remarks

Priority affects the order in which the server delivers messages to consumers (higher values first).

The JMS specification defines ten levels of priority value, from zero (lowest priority) to 9 (highest priority). The specification suggests that clients consider 0–4 as gradations of normal priority, and priorities 5–9 as gradations of expedited priority.

Programs can use this property to define a default priority for messages that this producer sends. Individual sending calls can override this default value.

## tibemsMsgProducer\_SetTimeToLive

*Function*

### Purpose

Set the time-to-live property of a producer object.

### C Declaration

```
tibems_status tibemsMsgProducer_SetTimeToLive(
    tibemsMsgProducer msgProducer,
    tibems_long timeToLive );
```

### COBOL Call

```
CALL "tibemsMsgProducer_SetTimeToLive"
    USING BY VALUE msgProducer,
          BY VALUE timeToLive,
          RETURNING tibems-status
END-CALL.
```



msgProducer has usage pointer.

## Parameters

Parameter	Description
msgProducer	Set the property of this producer.
timeToLive	Set the property to this value.

## Remarks

Time-to-live (in milliseconds) determines the expiration time of a message.

- If the time-to-live is non-zero, the expiration is the sum of that time-to-live and the sending client's current time (GMT). This rule applies even within sessions with transaction semantics—the timer begins with the send call, not the commit call.

- If the time-to-live is zero, then expiration is also zero—indicating that the message never expires. Programs can use this property to define a default time-to-live for messages that this producer sends. Individual sending calls can override this default value.

Whenever your application uses non-zero values for message expiration or time-to-live, you must ensure that clocks are synchronized among all the host computers that send and receive messages. Synchronize clocks to a tolerance that is a very small fraction of the smallest or time-to-live.

# Requestor

Requestors implement convenience functions for request-reply semantics. They send messages (called *requests*) and wait for *reply* messages in response.

## tibemsMsgRequestor

*Type*

### Purpose

Encapsulate request-reply semantics.

### Remarks

We recommend that programs follow these steps:

1. Create a `tibemsSession`, and use it to create either a `tibemsQueue` or `tibemsTopic` (respectively) for requests and replies.
2. Supply that session and destination to `tibemsMsgRequestor_Create` to create a `tibemsMsgRequestor`.
3. Call `tibemsMsgRequestor_Request` to send a request and receive a reply. You may repeat this step for several request and reply pairs.
4. Close the requestor object. `tibemsMsgRequestor_Close` also closes the requestor's session as a side effect.

Function	Description
<code>tibemsMsgRequestor_Close</code>	Close a message requestor.
<code>tibemsMsgRequestor_Create</code>	Create a message requestor.
<code>tibemsMsgRequestor_Request</code>	Send a request message; wait for a reply.

### See Also

`tibemsSession`

`tibemsQueue`

`tibemsTopic`

## tibemsMsgRequestor\_Close

*Function*

### Purpose

Close a message requestor.

### C Declaration

```
tibems_status tibemsMsgRequestor_Close(
    tibemsMsgRequestor msgRequestor );
```

## COBOL Call

```
CALL "tibemsMsgRequestor_Close"
    USING BY VALUE msgRequestor,
          RETURNING tibems-status
END-CALL.
```



msgRequestor has usage pointer.

### Parameters

Parameter	Description
msgRequestor	Close this requestor.

### Remarks

Also close the requestor's session as a side effect.

## tibemsMsgRequestor\_Create

*Function*

### Purpose

Create a message requestor.

### C Declaration

```
tibems_status tibemsMsgRequestor_Create(
    tibemsSession session,
    tibemsMsgRequestor* msgRequestor,
    tibemsDestination destination);
```

## COBOL Call

```
CALL "tibemsMsgRequestor_Create"
    USING BY VALUE session,
          BY REFERENCE msgRequestor,
          BY VALUE destination,
          RETURNING tibems-status
END-CALL.
```



session, msgRequestor, and destination have usage pointer.

### Parameters

Parameter	Description
session	The requestor operates within this session.  This session must not use transaction semantics. Its delivery mode must be either <a href="#">TIBEMS_AUTO_ACKNOWLEDGE</a> or <a href="#">TIBEMS_DUPS_OK_ACKNOWLEDGE</a> .
msgRequestor	The function stores the new requestor in this location.

Parameter	Description
destination	<p>The requestor sends request messages to this destination, and waits for replies on an internally created temporary destination.</p> <p>If the destination that request messages are sent to is a queue, then an internal temporary queue is created and used. If the destination that request messages are sent to is a topic, then an internal temporary topic is created and used.</p> <p>You <i>must</i> create this destination using the session you supply as the first argument.</p>

### Remarks

We recommend that programs follow these steps:

1. Create a `tibemsSession`, and use it to create a `tibemsQueue` or `tibemsTopic` for requests and replies.
2. Create a `tibemsMsgRequestor`, using the session and destination as arguments.
3. Send a request and receive a reply with `tibemsMsgRequestor_Request`. You may repeat this step for several request and reply pairs.
4. Close the requestor object. `tibemsMsgRequestor_Close` also closes the requestor's session as a side effect.

### See Also

[tibemsDestination](#)

[tibemsMsgRequestor\\_Close](#)

[tibemsMsgRequestor\\_Request](#)

[tibemsSession](#)

## tibemsMsgRequestor\_Request

*Function*

### Purpose

Send a request message; wait for a reply.

### C Declaration

```
tibems_status tibemsMsgRequestor_Request(
    tibemsMsgRequestor msgRequestor,
    tibemsMsg message,
    tibemsMsg* reply );
```

### COBOL Call

```
CALL "tibemsMsgRequestor_Request"
    USING BY VALUE msgRequestor,
         BY VALUE message,
         BY REFERENCE reply,
         RETURNING tibems-status
END-CALL.
```



`msgRequestor`, `message`, and `reply` have usage pointer.

**Parameters**

Parameter	Description
msgRequestor	Send and receive through this requestor.
message	Send this request message.
reply	When a reply message arrives, the function stores it in this location.

**Remarks**

This call blocks indefinitely, until a reply arrives.

The requestor receives only the first reply. It discards other replies that arrive subsequently.

# Connection

Connection objects represent a client program's network connection to the server.

## tibemsConnection

*Type*

### Purpose

Represent a server connection.

### Remarks

When a program first opens a connection, the connection is *stopped*—that is, it does not deliver inbound messages. To begin the flow of inbound messages, the program must explicitly call `tibemsConnection_Start`. (Outbound messages flow even before calling `tibemsConnection_Start`.)

The EMS C and COBOL APIs do *not* support the JMS methods `createConnectionConsumer` and `createDurableConnectionConsumer` (which are optional in the JMS specification).

### Asynchronous Errors

When a program uses a connection to send messages, the send calls can detect problems with the connection, and notify the client program (synchronously) by returning error codes.

However, when a program uses a connection only to receive messages, the client lacks that opportunity to detect problems. Instead, programs can handle such errors asynchronously by defining an exception listener callback (see `tibemsExceptionCallback`).

Function	Description
<code>tibemsConnection_Close</code>	Close the connection; reclaim resources.
<code>tibemsConnection_Create</code> <code>tibemsConnection_CreateSSL</code>	Create a new connection to an EMS server.
<code>tibemsConnection_CreateSession</code>	Create a session object.
<code>tibemsConnection_GetActiveURL</code>	Get the active URL of a connection.
<code>tibemsConnection_GetClientId</code>	Get the client ID of a connection.
<code>tibemsConnection_GetExceptionListener</code>	Get the exception listener of a connection.
<code>tibemsConnection_GetMetaData</code>	Get the metadata from a connection.
<code>tibemsConnection_IsDisconnected</code>	Check whether the connection has been disconnected.
<code>tibemsConnection_SetClientId</code>	Set the client ID of a connection.
<code>tibemsConnection_SetExceptionListener</code>	Set the exception listener for a connection.
<code>tibemsConnection_Start</code>	Start delivering inbound messages

Function	Description
<a href="#">tibemsConnection_Stop</a>	Stop delivering inbound messages.

## tibemsConnection\_Close

*Function*

### Purpose

Close the connection; reclaim resources.

### C Declaration

```
tibems_status tibemsConnection_Close(
    tibemsConnection connection );
```

### COBOL Call

```
CALL "tibemsConnection_Close"
    USING BY VALUE connection,
    RETURNING tibems-status
END-CALL.
```



connection has usage pointer.

### Parameters

Parameter	Description
connection	Close this connection.

### Remarks

Closing the connection is not sufficient to reclaim all of its resources; your program must explicitly close the sessions, producers, and consumers associated with the connection.

Closing a connection deletes all temporary destinations associated with the connection.

- **Blocking**

If any message listener or receive call associated with the connection is processing a message when the program calls this function, all facilities of the connection and its sessions remain available to those listeners until they return. In the meantime, this function blocks until that processing completes—that is, until all message listeners and receive calls have returned.

- **Acknowledge**

Closing a connection does *not* force acknowledgment in client-acknowledged sessions. When the program still has a message that it received from a connection that has since closed, [tibemsMsg\\_Acknowledge](#) indicates status code [TIBEMS\\_ILLEGAL\\_STATE](#).

- **Transactions**

Closing a connection rolls back all open transactions in all sessions associated with the connection.

Status Code	Description
<a href="#">TIBEMS_ILLEGAL_STATE</a>	The connection is currently processing a message callback.

**See Also**[tibemsMsg\\_Acknowledge](#)[tibemsMsgConsumer](#)[tibemsMsgProducer](#)[tibemsDestination](#)[tibemsSession](#)**tibemsConnection\_Create***Function***Purpose**

Create a new connection to an EMS server.

**C Declarations**

```

tibems_status tibemsConnection_Create(
    tibemsConnection* connection,
    const char* brokerURL,
    const char* clientId,
    const char* username,
    const char* password );

tibems_status tibemsConnection_CreateSSL(
    tibemsConnection* connection,
    const char* brokerURL,
    const char* clientId,
    const char* username,
    const char* password,
    tibemsSSLParams sslParams,
    const char* pk_password );

```

**COBOL Call**

```

CALL "tibemsConnection_Create"
    USING BY REFERENCE connection,
          BY REFERENCE brokerURL,
          BY REFERENCE clientId,
          BY REFERENCE username,
          BY REFERENCE password
    RETURNING tibems-status
END-CALL.

```

```

CALL "tibemsConnection_CreateSSL"
    USING BY REFERENCE connection,
          BY REFERENCE brokerURL,
          BY REFERENCE clientId,
          BY REFERENCE username,
          BY REFERENCE password,
          BY VALUE tibemsSSLParams,
          BY REFERENCE pk-password,
    RETURNING tibems-status
END-CALL.

```



connection and sslParams have usage pointer.

On IBM z/OS systems, the pk-password must always be a null value.

## Parameters

Parameter	Description
connection	Store the new connection in this location.
brokerURL	Find the EMS server at this URL. If configuring a fault-tolerant client, enter two or more URLs, as described in <a href="#">Configuring C and COBOL Clients for Fault-Tolerant Connections</a> .
clientId	Identify the client program to the server with this unique ID.
username	Authenticate the client program to the server with this user name.
password	Authenticate the client program to the server with this password.
sslParams	Establish SSL communication using these parameters.
pk_password	Private key password for SSL.

## Remarks

When the authentication parameters are null, the connection object presents a default user identity. If the server configuration permits that anonymous user, then the call succeeds.

Status Code	Description
<a href="#">TIBEMS_SERVER_NOT_CONNECTED</a>	<ul style="list-style-type: none"> <li>No server is running at the specified URL.</li> <li>The call could not communicate with a server because of mismatched SSL and TCP protocols.</li> <li>Other error situations are possible.</li> </ul>
<a href="#">TIBEMS_SECURITY_EXCEPTION</a>	<ul style="list-style-type: none"> <li>The server rejected the connection because the username or password was invalid.</li> <li>SSL setup is incorrect.</li> </ul>
<a href="#">TIBEMS_INVALID_CLIENT_ID</a>	The client ID is not unique; that is, another client already uses the ID.

## See Also

[tibemsConnectionFactory\\_Create](#)

[SSL Implementation on IBM EBCDIC Systems](#)

## **tibemsConnection\_CreateSession**

Function

### Purpose

Create a session object.

## C Declaration

```
tibems_status tibemsConnection_CreateSession(
    tibemsConnection connection,
    tibemsSession* session,
    tibems_bool transacted,
    tibemsAcknowledgeMode acknowledgeMode );
```

## COBOL Call

```
CALL "tibemsConnection_CreateSession"
    USING BY VALUE connection,
          BY REFERENCE session,
          BY VALUE transacted,
          BY VALUE acknowledgeMode,
          RETURNING tibems-status
END-CALL.
```



connection and session have usage pointer.

## Parameters

Parameter	Description
connection	Create a session on this connection.
session	Store the new session in this location.
transacted	When true, the new session has transaction semantics. When false, it has non-transaction semantics.
acknowledgeMode	Determine the acknowledge mode of the session. Supply a value enumerated by <a href="#">tibemsAcknowledgeMode</a> .

## Remarks

The new session uses the connection for all server communications.

## See Also

[tibemsMsg\\_Acknowledge](#)

[tibemsSession](#)

[tibemsAcknowledgeMode](#)

## tibemsConnection\_GetActiveURL

*Function*

### Purpose

Get the active URL of a connection.

### C Declaration

```
tibems_status tibemsConnection_GetActiveURL(
    tibemsConnection connection,
    char** serverURL );
```

## COBOL Call

```
CALL "tibemsConnection_GetActiveURL"
    USING BY VALUE connection,
          BY REFERENCE serverURL,
          RETURNING tibems-status
END-CALL.
```



connection and serverURL have usage pointer.

### Parameters

Parameter	Description
connection	Get the active URL of this connection.
serverURL	Store a pointer to the URL in this location.

### Remarks

This property is the URL of the server at the other endpoint of the connection. When the connection interacts with several servers in a fault-tolerant arrangement, this property indicates the current active server.

## tibemsConnection\_GetClientId

*Function*

### Purpose

Get the client ID of a connection.

### C Declaration

```
tibems_status tibemsConnection_GetClientId(
    tibemsConnection connection,
    const char** clientId );
```

## COBOL Call

```
CALL "tibemsConnection_GetClientId"
    USING BY VALUE connection,
          BY REFERENCE clientId,
          RETURNING tibems-status
END-CALL.
```



connection and clientId have usage pointer.

### Parameters

Parameter	Description
connection	Get the client ID of this connection.
clientId	Store a pointer to the ID string in this location.

### Remarks

Each connection uses a unique client ID.

Client IDs partition the namespace of durable subscribers; see [tibemsSession\\_CreateDurableSubscriber](#).

### See Also

[tibemsConnection\\_SetClientId](#)

## tibemsConnection\_GetExceptionListener

*Function*

### Purpose

Get the exception listener of a connection.

### C Declaration

```
tibems_status tibemsConnection_GetExceptionListener(
    tibemsConnection connection,
    tibemsExceptionCallback* listener,
    void** closure );
```

### Parameters

Parameter	Description
connection	Get the exception listener of this connection.
listener	Store a pointer to the exception listener callback in this location.
closure	Store a pointer to the exception listener closure argument in this location.

### Remarks

This is an alternate pathway for alerting a client program of connection problems. The program defines an exception listener callback function, and registers the callback using [tibemsConnection\\_SetExceptionListener](#). When the client library detects a connection problem, it calls the callback with an exception argument that details the problem.

This call is not available in COBOL.

### See Also

[tibemsConnection\\_SetExceptionListener](#)

[Asynchronous Errors](#).

[tibemsExceptionCallback](#)

## tibemsConnection\_GetMetaData

*Function*

### Purpose

Get the metadata from a connection.

### C Declaration

```
tibems_status tibemsConnection_GetMetaData(
    tibemsConnection connection,
    tibemsConnectionMetaData* metaData );
```

## COBOL Call

```
CALL "tibemsConnection_GetMetaData"
    USING BY VALUE connection,
          BY REFERENCE metaData,
          RETURNING tibems-status
END-CALL.
```



connection and metaData have usage pointer.

### Parameters

Parameter	Description
connection	Get the metadata of this connection.
metaData	Store a metadata object in this location.

### See Also

[tibemsConnectionMetaData](#)

## tibemsConnection\_IsDisconnected

*Function*

### Purpose

Check whether the connection has been disconnected.

### C Declaration

```
tibems_status tibemsConnection_IsDisconnected(
    tibemsConnection connection,
    tibems_bool* disconnected);
```

## COBOL Call

```
CALL "tibemsConnection_IsDisconnected"
    USING BY VALUE connection,
          BY REFERENCE disconnected,
          RETURNING tibems-status
END-CALL.
```



connection has usage pointer.

### Parameters

Parameter	Description
connection	Get the status of this connection.
disconnected	Store the connection status in this location.

### Remarks

This function gets a value indicating whether the connection is disconnected from the server. If the connection is disconnected from the server, `tibemsConnection_IsDisconnected` sets the connection status to TRUE. Otherwise, the connection status is FALSE.

That is, if the client has called `tibemsConnection_Close`, or if the connection has been terminated due to a network failure, `tibemsConnection_IsDisconnected` returns TRUE.

## `tibemsConnection_SetClientId`

*Function*

### Purpose

Set the client ID of a connection.

### C Declaration

```
tibems_status tibemsConnection_SetClientId(
    tibemsConnection connection,
    const char* clientId );
```

### COBOL Call

```
CALL "tibemsConnection_SetClientId"
    USING BY VALUE connection,
         BY REFERENCE clientId,
         RETURNING tibems-status
END-CALL.
```



`connection` has usage pointer.

### Parameters

Parameter	Description
<code>connection</code>	Set the client ID of this connection.
<code>clientId</code>	Set the client ID to this string.

### Remarks

Each connection uses a unique client ID.

Client IDs partition the namespace of durable subscribers; see [tibemsSession\\_CreateDurableSubscriber](#).

Administrators can configure connection factories to assign client IDs to new connections. Alternatively, administrators can allow client programs to assign their own IDs. If the factory does not assign an ID, the program may set this property. However, it is illegal to overwrite an existing client ID value, and or to set this property after using the connection in any way (for example, after creating a session, or starting the connection); attempting to set this property in these situations results in `TIBEMS_ILLEGAL_STATE`.

### See Also

[tibemsConnection\\_GetClientId](#)

## `tibemsConnection_SetExceptionListener`

*Function*

### Purpose

Set the exception listener for a connection.

## C Declaration

```
tibems_status tibemsConnection_SetExceptionListener(
    tibemsConnection connection,
    tibemsExceptionCallback listener,
    const void* closure );
```

## COBOL Call

```
CALL "tibemsConnection_SetExceptionListener_STL"
    USING BY VALUE      tibemsConnection,
           BY REFERENCE tibems-Exception-Status,
           BY VALUE     TIBEMS-NULLPTR,
           RETURNING    tibems-status
END-CALL.
```



connection has usage pointer.

tibems-Exception-Status has usage binary.

## Parameters

Parameter	Description
connection	Set the exception listener for this connection.
listener	Register this exception listener callback.
closure	Register this closure argument.

## Remarks

This is an alternate pathway for alerting a client program of connection problems. The program defines an exception listener callback function, and calls this function to register the callback and a closure argument. When the client library detects a connection problem, it calls the callback with a status code that identifies the problem.

## See Also

[tibemsConnection\\_GetExceptionListener](#)

[Asynchronous Errors](#).

[tibemsExceptionCallback](#)

## tibemsConnection\_Start

*Function*

### Purpose

Start delivering inbound messages.

### C Declaration

```
tibems_status tibemsConnection_Start(
    tibemsConnection connection );
```

### COBOL Call

```
CALL "tibemsConnection_Start"
    USING BY VALUE connection,
```

RETURNING tibems-status  
END-CALL.



connection has usage pointer.

### Parameters

Parameter	Description
connection	Start delivering inbound messages on this connection.

### Remarks

When a connection is created, it is stopped. It does not deliver inbound messages until the program calls this function to explicitly start it.

If the connection is not stopped, this call has no effect.

Outbound messages flow even before calling start.

### See Also

[tibemsConnection\\_Stop](#)

## tibemsConnection\_Stop

*Function*

### Purpose

Stop delivering inbound messages.

### C Declaration

```
tibems_status tibemsConnection_Stop(  
    tibemsConnection connection );
```

### COBOL Call

```
CALL "tibemsConnection_Stop"  
    USING BY VALUE connection,  
    RETURNING tibems-status  
END-CALL.
```



connection has usage pointer.

### Parameters

Parameter	Description
connection	Stop delivering inbound messages on this connection.

### Remarks

This call temporarily stops the connection from delivering inbound messages. A program can restart delivery by calling [tibemsConnection\\_Start](#).

When a connection is created, it is stopped. It does not deliver inbound messages until the program calls this function to explicitly start it.

If the connection is already stopped, this call has no effect.

- **Effect**

When this call returns, the connection has stopped delivery to all consumers associated with the connection:

- Messages do not arrive to trigger asynchronous message handler events, nor message listeners.
- Synchronous receive functions block. If their timeout intervals expire, they return null.

- **Blocking**

If any message listener or receive call associated with the connection is processing a message when the program calls this function, all facilities of the connection and its sessions remain available to those listeners until they return. In the meantime, this function blocks until that processing completes—that is, until all message listeners and receive calls have returned.

However, the stopped connection prevents the client program from processing any new messages.

- **Sending**

A stopped connection can still send outbound messages.

**See Also**

[tibemsConnection\\_Start](#)

## tibemsConnectionMetaData

*Type*

**Purpose**

Get information about EMS and the EMS provider.

**Remarks**

Functions of type `tibemsConnectionMetaData` retrieve information about the EMS application.

Function	Description
<a href="#">tibemsConnectionMetaData_GetEMSMajorVersion</a> <a href="#">tibemsConnectionMetaData_GetEMSMinorVersion</a> <a href="#">tibemsConnectionMetaData_GetEMSVersion</a>	Get information about the JMS specification supported by EMS.
<a href="#">tibemsConnectionMetaData_GetProviderMajorVersion</a> <a href="#">tibemsConnectionMetaData_GetProviderMinorVersion</a> <a href="#">tibemsConnectionMetaData_GetProviderVersion</a>	Get the version number of the EMS installation.
<a href="#">tibemsConnectionMetaData_GetEMSProviderName</a>	Get the name of the EMS provider.

**See Also**

[tibemsConnection\\_GetMetaData](#)

## tibemsConnectionMetaData\_GetEMS

Function

### Purpose

Get information about the JMS specification supported by EMS.

### C Declaration

```
tibems_status tibemsConnectionMetaData_GetEMSMajorVersion(
    tibemsConnectionMetaData metaData,
    tibems_int* majorVersion);

tibems_status tibemsConnectionMetaData_GetEMSMinorVersion(
    tibemsConnectionMetaData metaData,
    tibems_int* minorVersion);

tibems_status tibemsConnectionMetaData_GetEMSVersion(
    tibemsConnectionMetaData metaData,
    const char** version);
```

### COBOL Call

```
CALL "tibemsConnectionMetaData_GetEMSMajorVersion"
    USING BY VALUE metaData
         BY REFERENCE majorVersion
         RETURNING tibems_status
END-CALL.

CALL "tibemsConnectionMetaData_GetEMSMinorVersion"
    USING BY VALUE metaData
         BY REFERENCE minorVersion
         RETURNING tibems_status
END-CALL.

CALL "tibemsConnectionMetaData_GetEMSVersion"
    USING BY VALUE metaData
         BY REFERENCE version
         RETURNING tibems_status
END-CALL.
```



metadata has usage pointer.

### Parameters

Parameter	Description
metaData	The tibemsConnectionMetaData object from which the version number will be extracted.
majorVersion	tibemsConnectionMetaData_GetEMSMajorVersion uses this location to store the major version number of the JMS specification supported by the EMS application.
minorVersion	tibemsConnectionMetaData_GetEMSMinorVersion uses this location to store the minor version number of the JMS specification supported by the EMS application.
version	tibemsConnectionMetaData_GetEMSVersion uses this location to store the version number of the JMS specification supported by the EMS application.

## Remarks

EMS applications can retrieve the metaData object from any connection; see [tibemsConnection\\_GetMetaData](#).

## tibemsConnectionMetaData\_GetProvider

Function

### Purpose

Get the version number of the EMS installation.

### C Declaration

```
tibems_status tibemsConnectionMetaData_GetProviderMajorVersion(
    tibemsConnectionMetaData metaData,
    tibems_int* providerMajorVersion);

tibems_status tibemsConnectionMetaData_GetProviderMinorVersion(
    tibemsConnectionMetaData metaData,
    tibems_int* providerMinorVersion);

tibems_status tibemsConnectionMetaData_GetProviderVersion(
    tibemsConnectionMetaData metaData,
    const char** providerVersion);
```

### COBOL Call

```
CALL "tibemsConnectionMetaData_GetProviderMajorVersion"
    USING BY VALUE metaData
         BY REFERENCE majorVersion
         RETURNING tibems_status
END-CALL.

CALL "tibemsConnectionMetaData_GetProviderMinorVersion"
    USING BY VALUE metaData
         BY REFERENCE minorVersion
         RETURNING tibems_status
END-CALL.

CALL "tibemsConnectionMetaData_GetProviderVersion"
    USING BY VALUE metaData
         BY REFERENCE version
         RETURNING tibems_status
END-CALL.
```



metaData and version have usage pointers.

### Parameters

Parameter	Description
metaData	tibemsConnectionMetaData object from which the version number will be extracted.
providerMajorVersion	tibemsConnectionMetaData_GetProviderMajorVersion uses this location to store the major version number of the EMS application.

Parameter	Description
providerMinorVersion	tibemsConnectionMetaData_GetProviderMinorVersion uses this location to store the minor version number of the EMS application.
providerVersion	tibemsConnectionMetaData_GetProviderVersion uses this location to store the version number of the EMS application.

### Remarks

EMS applications can retrieve the metaData object from any connection; see [tibemsConnection\\_GetMetaData](#).

## tibemsConnectionMetaData\_GetEMSProviderName

Function

### Purpose

Get the name of the EMS provider.

### C Declaration

```
tibems_status tibemsConnectionMetaData_GetEMSProviderName(
    tibemsConnectionMetaData metaData,
    const char** providerName);
```

### COBOL Call

```
CALL "tibemsConnectionMetaData_GetEMSProviderName"
    USING BY VALUE metaData
         BY REFERENCE providerName
         RETURNING tibems_status
END-CALL.
```



metaData has usage pointer.

### Parameters

Parameter	Description
metaData	The object from which the provider name will be extracted.
providerName	The location where the provider's vendor name will be stored.

### Remarks

tibemsConnectionMetaData\_GetEMSProviderName provides the name of the EMS provider, TIBCO Software Inc.

EMS applications can retrieve the metaData object from any connection; see [tibemsConnection\\_GetMetaData](#).

## tibemsExceptionCallback

Type

### Purpose

Programs define functions of this type to asynchronously detect problems with connections.

### C Declaration

```
typedef void (*tibemsExceptionCallback) (
    tibemsConnection connection,
    tibems_status status,
    void* closure);
```

### Parameters

Parameter	Description
connection	Receive the connection object.
status	Receive a status code, which identifies the connection problem.
closure	Receive the closure data, which the program supplied in the call that registered the callback.

### Remarks

When a program uses a connection to send messages, the send calls can detect problems with the connection, and notify the client program by returning an error status code. However, when a program uses a connection only to receive messages, the client cannot detect errors in this way.

This callback provides an alternate pathway for alerting a client program of connection problems. The program implements this callback, and registers it with the connection object. When the client library detects a connection problem, it calls this callback with a status code that identifies the problem.

This call is not supported in COBOL.

### See Also

[tibemsConnection](#)

[tibemsConnection\\_SetExceptionListener](#)

## tibemsMulticastExceptionCallback

Function Type



Along with the multicast feature, this function was deprecated in software release 8.3.0 and will no longer be supported in future releases.

### Purpose

Programs define functions of this type to asynchronously detect conditions with EMS multicast that may affect message consumers.

### C Declaration

```
typedef void (*tibemsMulticastExceptionCallback) (
    tibemsConnection connection,
```

```

tibemsSession session,
tibemsMsgConsumer consumer,
tibems_status status,
const char* description,
void* closure);

```

### Parameters

Parameter	Description
connection	Receive the connection object.
session	Receive the session object.
consumer	Receive the consumer object.
status	Receive a status code, which identifies the multicast problem.
description	Receive a text description describing the multicast problem.
closure	Receive the closure data, which the program supplied in the call that registered the callback.

### Remarks

When a program uses a multicast consumer to receive messages, EMS can detect conditions that may indicate a problem, and notify the client program by returning an error status code. The client application can then take the appropriate action.

This callback provides a pathway for alerting a client program of multicast problems. The program implements this callback and registers it. When the client library detects a multicast problem, it calls this callback and passes it the appropriate EMS objects, a status code that identifies the problem, and a detailed description of the problem.

This callback is invoked for each consumer that is affected by the multicast warning or error. Some applications may just simply log the problem; others may take further measures.

This call is not supported in COBOL, and is not supported on z/OS and IBM i systems.

### See Also

[tibems\\_SetMulticastExceptionListener](#)

## tibemsSSL

Type

### Functions

Function	Description
<a href="#">tibemsSSL_GetTrace</a>	Determine whether SSL tracing is enabled.
<a href="#">tibemsSSL_OpenSSLVersion</a>	Get a string representing the OpenSSL version number.
<a href="#">tibemsSSL_SetTrace</a>	Enable or disable SSL tracing.

*Certificate Encodings*

Constant	Value
TIBEMS_SSL_ENCODING_AUTO	(0x0000)
TIBEMS_SSL_ENCODING_PEM	(0x0001)
TIBEMS_SSL_ENCODING_DER	(0x0002)
TIBEMS_SSL_ENCODING_BER	(0x0004)
TIBEMS_SSL_ENCODING_PKCS7	(0x0010)
TIBEMS_SSL_ENCODING_PKCS8	(0x0020)
TIBEMS_SSL_ENCODING_PKCS12	(0x0040)
TIBEMS_SSL_ENCODING_ENTRUST	(0x0100)
TIBEMS_SSL_ENCODING_KEYSTORE	(0x0200)

**tibemsSSL\_GetTrace**

Function

**Purpose**

Determine whether SSL tracing is enabled.

**C Declaration**

```
tibems_bool tibemsSSL_GetTrace(void);
tibems_bool tibemsSSL_GetDebugTrace(void);
```

**IBM Systems**

This function is not supported on z/OS and IBM i systems. For more information, see [SSL Implementation on IBM EBCDIC Systems](#).

**Remarks**

Two levels of SSL tracing are available—regular tracing and debug tracing (more detailed).

If tracing is enabled, these calls return TIBEMS\_TRUE.

If tracing is disabled, they return TIBEMS\_FALSE.

**tibemsSSL\_OpenSSLVersion**

Function

**Purpose**

Get a string representing the OpenSSL version number.

## C Declaration

```
const char* tibemsSSL_OpenSSLVersion(
    char* buffer,
    tibems_int buf_size );
```

## IBM Systems

This function is not supported on z/OS and IBM i systems. For more information, see [SSL Implementation on IBM EBCDIC Systems](#).

## Parameters

Parameter	Description
buffer	The function copies the version string in this buffer.
buf_size	Length (in bytes) of the buffer.

## Remarks

The versions string has the format *major.minor.update*.

A null character terminates the version string.

## tibemsSSL\_SetTrace

Function

## Purpose

Enable or disable SSL tracing.

## C Declaration

```
void tibemsSSL_SetTrace(
    tibems_bool trace );

void tibemsSSL_SetDebugTrace(
    tibems_bool trace );
```

## IBM Systems

These functions are not supported on IBM z/OS systems. For more information, see [SSL Implementation on IBM EBCDIC Systems](#).

## Parameters

Parameter	Description
trace	TIBEMS_TRUE enables tracing. TIBEMS_FALSE disables tracing.

## Remarks

Two levels of SSL tracing are available—regular tracing and debug tracing (more detailed).

## tibemsSSLParams

Type

### Purpose

Group parameters representing a client identity.

### Remarks

These parameters apply when creating SSL connections to the EMS server. On most systems, secure connections are created using OpenSSL. Users on IBM z/OS systems must create SSL connections using IBM System SSL. For more information, see [SSL Implementation on IBM EBCDIC Systems](#).

Function	Description
<a href="#">tibemsSSLParams_AddIssuerCert</a>	Add one or more issuer certificates to the SSL parameter object.
<a href="#">tibemsSSLParams_AddTrustedCert</a>	Add one or more trusted certificates to the SSL parameter object.
<a href="#">tibemsSSLParams_Create</a>	Create a new SSL parameter object.
<a href="#">tibemsSSLParams_Destroy</a>	Destroy an SSL parameter object.
<a href="#">tibemsSSLParams_GetIdentity</a>	Get the client identity that an SSL parameter object represents.
<a href="#">tibemsSSLParams_GetPrivateKey</a>	Get the private key from an SSL parameter object.
<a href="#">tibemsSSLParams_SetAuthOnly</a>	Set client connections to use SSL only during initial connection authentication.
<a href="#">tibemsSSLParams_SetCiphers</a>	Set the cipher suites for SSL connections.
<a href="#">tibemsSSLParams_SetExpectedHostName</a>	Set the expected host name.
<a href="#">tibemsSSLParams_SetHostNameVerifier</a>	Set the host name verifier function.
<a href="#">tibemsSSLParams_SetIdentity</a>	Set the identity of the client program.
<a href="#">tibemsSSLParams_SetPrivateKey</a>	Set the client's private key.
<a href="#">tibemsSSLParams_SetVerifyHost</a>	Set flags that enable client verification of the host certificate or host name.

### tibemsSSLParams\_AddIssuerCert

Function

### Purpose

Add one or more issuer certificates to the SSL parameter object.

## C Declaration

```
tibems_status tibemsSSLParams_AddIssuerCert(
    tibemsSSLParams SSLParams,
    const void* data,
    tibems_int size,
    tibems_int encoding );

tibems_status tibemsSSLParams_AddIssuerCertFile(
    tibemsSSLParams SSLParams,
    const char* filename,
    tibems_int encoding );
```

## IBM Systems

These functions are not supported on z/OS and IBM i systems. For more information, see [SSL Implementation on IBM EBCDIC Systems](#).

## Parameters

Parameter	Description
SSLParams	Add the certificates to this SSL parameter object.
data	Use the certificate data at this location.
size	Length of the certificate data (in bytes).
encoding	Interpret the certificate data using this encoding; for values, see <a href="#">Certificate Encodings</a> .
filename	Read the certificate data from this file.

## Remarks

Issuer certificates are certificates that authenticate the client's certificate; the certificate authority (CA) that issued the client's certificate supplies these. SSL clients must supply them during the SSL handshake, so your program must set them.

If the parameter object already has issuer certificates, this call adds to that set; it does not overwrite them.

## tibemsSSLParams\_AddTrustedCert

Function

### Purpose

Add one or more trusted certificates to the SSL parameter object.

## C Declaration

```
tibems_status tibemsSSLParams_AddTrustedCert(
    tibemsSSLParams SSLParams,
    const void* data,
    tibems_int size,
    tibems_int encoding );

tibems_status tibemsSSLParams_AddTrustedCertFile(
    tibemsSSLParams SSLParams,
```

```
const char* filename,
tibems_int encoding );
```

## IBM Systems

These functions are not supported on z/OS and IBM i systems. For more information, see [SSL Implementation on IBM EBCDIC Systems](#).

## Parameters

Parameter	Description
SSLParams	Add the certificates to this SSL parameter object.
data	Use the certificate data at this location.
size	Length of the certificate data (in bytes).
encoding	Interpret the certificate data using this encoding; for values, see <a href="#">Certificate Encodings</a> .
filename	Read the certificate data from this file.

## Remarks

Trusted certificates are certificates that authenticate the server's certificate; the certificate authority (CA) that issued the server's certificate supplies these. SSL clients may verify them during the SSL handshake; if your program verifies host certificates (see [tibemsSSLParams\\_SetVerifyHost](#)), then you must register trusted certificates as well.

If the parameter object already has trusted certificates, this call adds to that set; it does not overwrite them.

## tibemsSSLParams\_Create

Function

### Purpose

Create a new SSL parameter object.

### C Declaration

```
tibemsSSLParams tibemsSSLParams_Create(void);
```

## IBM Systems

This function is not supported on z/OS and IBM i systems. See [SSL Implementation on IBM EBCDIC Systems](#).

## tibemsSSLParams\_Destroy

Function

### Purpose

Destroy an SSL parameter object.

## C Declaration

```
void tibemsSSLParams_Destroy(
    tibemsSSLParams SSLParams );
```

## IBM Systems

This function is not supported on z/OS and IBM i systems. For more information, see [SSL Implementation on IBM EBCDIC Systems](#).

## Parameters

Parameter	Description
SSLParams	Destroy this SSL parameter object.

## tibemsSSLParams\_GetIdentity

Function

## Purpose

Get the client identity that an SSL parameter object represents.

## C Declaration

```
tibems_status tibemsSSLParams_GetIdentity(
    tibemsSSLParams SSLParams,
    const void** data,
    tibems_int* size,
    tibems_int* encoding );
```

## IBM Systems

This function is not supported on z/OS and IBM i systems. For more information, see [SSL Implementation on IBM EBCDIC Systems](#).

## Parameters

Parameter	Description
SSLParams	Get the identity from this SSL parameter object.
data	The function stores in this location a pointer to the identity data within the <a href="#">tibemsSSLParams</a> object.
size	The function stores the length (in bytes) of the identity data in this location.
encoding	The function stores the encoding of the identity data in this location; for values, see <a href="#">Certificate Encodings</a> .

## Remarks

A client identity includes a certificate and private key; it may also include issuer certificates (optional).

## tibemsSSLParams\_GetPrivateKey

Function

### Purpose

Get the private key from an SSL parameter object.

### C Declaration

```
tibems_status tibemsSSLParams_GetPrivateKey(
    tibemsSSLParams SSLParams,
    const void** data,
    tibems_int* size,
    tibems_int* encoding );
```

### IBM Systems

This function is not supported on z/OS and IBM i systems. For more information, see [SSL Implementation on IBM EBCDIC Systems](#).

### Parameters

Parameter	Description
SSLParams	Get the private key from this SSL parameter object.
data	The function stores in this location a pointer to the key data within the <a href="#">tibemsSSLParams</a> object.
size	The function stores the length (in bytes) of the key data in this location.
encoding	The function stores the encoding of the key data in this location; for values, see <a href="#">Certificate Encodings</a> .

## tibemsSSLParams\_SetAuthOnly

Function

### Purpose

Set client connections to use SSL only during initial connection authentication.

### C Declaration

```
tibems_status tibemsSSLParams_SetAuthOnly(
    tibemsSSLParams SSLParams,
    tibems_bool auth_only );
```

### IBM Systems

This function is not supported on z/OS and IBM i systems. For more information, see [SSL Implementation on IBM EBCDIC Systems](#).

### Parameters

Parameter	Description
SSLParams	Set the value in this SSL parameter object.

Parameter	Description
auth_only	TIBEMS_TRUE instructs the SSL parameter object to request a connection that uses SSL only for authentication.  TIBEMS_FALSE instructs the SSL parameter object to request a connection that uses SSL to secure all data.

### Remarks

If `auth_only` is `TIBEMS_TRUE`, connections use SSL only for authentication and switch to the TCP protocol for all subsequent messaging. If not, by default, SSL is used for the lifetime of a connection.

For background information, see [SSL Authentication Only](#) in *TIBCO Enterprise Message Service User's Guide*.

## tibemsSSLParams\_SetCiphers

Function

### Purpose

Set the cipher suites for SSL connections.

### C Declaration

```
tibems_status tibemsSSLParams_SetCiphers(
    tibemsSSLParams SSLParams,
    const char* ciphers );
```

### IBM Systems

This function is not supported on z/OS and IBM i systems. For more information, see [SSL Implementation on IBM EBCDIC Systems](#).

### Parameters

Parameter	Description
SSLParams	Set the value in this SSL parameter object.
ciphers	Specify the cipher suites that the client can use.  Supply a colon-separated list of cipher names. Names may be either OpenSSL names, or longer descriptive names.

## tibemsSSLParams\_SetExpectedHostName

Function

### Purpose

Set the expected host name.

### C Declaration

```
tibems_status tibemsSSLParams_SetExpectedHostName(
    tibemsSSLParams SSLParams,
    const char* expected_hostname );
```

## IBM Systems

This function is not supported on z/OS and IBM i systems. For more information, see [SSL Implementation on IBM EBCDIC Systems](#).

### Parameters

Parameter	Description
SSLParams	Set the value in this SSL parameter object.
expected_hostname	Use this value.

### Remarks

This parameter applies when establishing an SSL connection to the EMS server. If host name verification is enabled, an application-specific verifier function checks that the actual host name where the server is running is the same as this expected host name.

### See Also

[tibemsSSLParams\\_SetHostNameVerifier](#)

[tibemsSSLParams\\_SetVerifyHost](#)

[tibemsSSLHostNameVerifier](#)

## tibemsSSLParams\_SetHostNameVerifier

Function

### Purpose

Set the host name verifier function.

### C Declaration

```
tibems_status tibemsSSLParams_SetHostNameVerifier(
    tibemsSSLParams SSLParams,
    tibemsSSLHostNameVerifier verifier,
    const void* closure );
```

## IBM Systems

This function is not supported on z/OS and IBM i systems. For more information, see [SSL Implementation on IBM EBCDIC Systems](#).

### Parameters

Parameter	Description
SSLParams	Set the value in this SSL parameter object.
verifier	Use this verifier function.
closure	Supply application-specific data. Each call to the verifier function passes this data as an argument.

## Remarks

When creating a connection to the EMS server, an application-specific verifier function checks that the actual host name where the server is running is the same as this expected host name.

## See Also

[tibemsSSLParams\\_SetExpectedHostName](#)

[tibemsSSLParams\\_SetVerifyHost](#)

[tibemsSSLHostNameVerifier](#)

## tibemsSSLParams\_SetIdentity

Function

### Purpose

Set the identity of the client program.

### C Declaration

```
tibems_status tibemsSSLParams_SetIdentity(
    tibemsSSLParams SSLParams,
    const void* data,
    tibems_int size,
    tibems_int encoding );

tibems_status tibemsSSLParams_SetIdentityFile(
    tibemsSSLParams SSLParams,
    const char* filename,
    tibems_int encoding );
```

### IBM Systems

These functions are not supported on z/OS and IBM i systems. For more information, see [SSL Implementation on IBM EBCDIC Systems](#).

### Parameters

Parameter	Description
SSLParams	Set the value in this SSL parameter object.
data	Data must include the client's certificate and private key. It may optionally include issuer certificates.
size	Supply the size (in bytes) of the data.
filename	Read identity data from this file.
encoding	Interpret the certificate data using this encoding; for values, see <a href="#">Certificate Encodings</a> .

## tibemsSSLParams\_SetPrivateKey

Function

### Purpose

Set the client's private key.

### C Declaration

```
tibems_status tibemsSSLParams_SetPrivateKey(
    tibemsSSLParams SSLParams,
    const void* data,
    tibems_int size,
    tibems_int encoding );

tibems_status tibemsSSLParams_SetPrivateKeyFile(
    tibemsSSLParams SSLParams,
    const char* filename,
    tibems_int encoding );
```

### IBM Systems

These functions are not supported on z/OS and IBM i systems. For more information, see [SSL Implementation on IBM EBCDIC Systems](#).

### Parameters

Parameter	Description
SSLParams	Set the value in this SSL parameter object.
data	Private key data.
size	Supply the size (in bytes) of the data.
filename	Read private key data from this file.
encoding	Interpret the data using this encoding; for values, see <a href="#">Certificate Encodings</a> .

## tibemsSSLParams\_SetVerifyHost

Function

### Purpose

Set flags that enable client verification of the host certificate or host name.

### C Declaration

```
tibems_status tibemsSSLParams_SetVerifyHost(
    tibemsSSLParams SSLParams,
    tibems_bool verify );

tibems_status tibemsSSLParams_SetVerifyHostName(
    tibemsSSLParams SSLParams,
    tibems_bool verify );
```

## IBM Systems

These functions are not supported on z/OS and IBM i systems. For more information, see [SSL Implementation on IBM EBCDIC Systems](#).

### Parameters

Parameter	Description
SSLParams	Set the value in this SSL parameter object.
verify	TIBEMS_TRUE enables verification. TIBEMS_FALSE disables verification.

### Remarks

Both of verification actions are enabled by default (unless a program explicitly disables them).

`tibemsSSLParams_SetVerifyHost` enables checking that the server host's certificate was signed by a trusted CA; see `tibemsSSLParams_AddTrustedCert`.

`tibemsSSLParams_SetVerifyHostName` enables checking the server's actual host name against an expected server host name; see `tibemsSSLParams_SetExpectedHostName`.

### See Also

[tibemsSSLParams\\_SetHostNameVerifier](#)

[tibemsSSLHostNameVerifier](#)

## tibemsSSLHostNameVerifier

Type

### Purpose

Programs define functions of this type to check server identity based on the server's host name.

### C Declaration

```
typedef tibems_status (*tibemsSSLHostNameVerifier) (
    const char* connected_hostname,
    const char* expected_hostname,
    const char* certificate_name,
    void* closure );
```

## IBM Systems

This function is not supported on z/OS and IBM i systems. For more information, see [SSL Implementation on IBM EBCDIC Systems](#).

### Parameters

Parameter	Description
connected_hostname	Receive the actual host name of the server to which the client program is attempting to connect.

Parameter	Description
<code>expected_hostname</code>	Receive the host name that the client expects the server to be running on.
<code>certificate_name</code>	Receive the host name in the server's public certificate.
<code>closure</code>	Receive application-specific data.

### Remarks

SSL attempts to verify that the EMS server hostname (taken from the server's certificate identity) matches the hostname in the server URL. Your program can use the default matching behavior, or customize it in different ways.

- The default behavior is a straightforward string comparison, matching the hostname from the server certificate against the hostname of the connected URL.
- If you set an expected hostname, then the match compares the hostname from the server certificate against the expected hostname (instead of the URL hostname).
- You may also define and set a hostname verifier function, which can override a string mismatch. If the string comparison fails, then SSL calls your verifier function to determine whether to accept the hostname anyway. Your function receives three hostnames—the connected name, the expected name, and the certificate hostname—and must return a status code indicating the final match result:
  - `TIBEMS_OK` indicates a successful check.
  - `TIBEMS_SECURITY_EXCEPTION` indicates a failed check.

### See Also

[tibemsSSLParams\\_SetExpectedHostName](#)

[tibemsSSLParams\\_SetHostNameVerifier](#)

[tibemsSSLParams\\_SetVerifyHost](#)

# Connection Factory

Connection factories let administrators preconfigure client connections to the EMS server.

## tibemsConnectionFactory

*Type*

### Purpose

Administered object for creating server connections.

### Remarks

Connection factories are administered objects. They support concurrent use.

Administrators define connection factories in a repository. Each connection factory has administrative parameters that guide the creation of server connections. Usage follows either of two models:

### EMS Server

You can use the EMS server as a name service provider—one `tibemspd` process provides both the name repository and the message service. Administrators define factories in the name repository. Client programs create connection factory objects with the URL of the repository, and call `tibemsConnectionFactory_CreateConnection`. This function automatically accesses the corresponding factory in the repository, and uses it to create a connection to the message service.

Function	Description
<code>tibemsConnectionFactory_Create</code>	Create a connection factory.
<code>tibemsConnectionFactory_CreateConnection</code>	Create a connection object.
<code>tibemsConnectionFactory_CreateXAConnection</code>	Create an XA connection object.
<code>tibemsConnectionFactory_Destroy</code>	Destroy a connection factory object.
<code>tibemsConnectionFactory_GetSSLProxyHost</code>	Get the SSL proxy host from a connection factory.
<code>tibemsConnectionFactory_GetSSLProxyPort</code>	Get the SSL proxy port from a connection factory.
<code>tibemsConnectionFactory_GetSSLProxyUser</code>	Get the SSL proxy username from a connection factory.
<code>tibemsConnectionFactory_GetSSLProxyPassword</code>	Get the SSL proxy password from a connection factory.
<code>tibemsConnectionFactory_Print</code>	Print the parameters set in a connection factory object.

Function	Description
<code>tibemsConnectionFactory_PrintToBuffer</code>	Print the parameters set in a connection factory object to a buffer.
<code>tibemsConnectionFactory_SetClientID</code>	Set the client ID of a connection factory object.
<code>tibemsConnectionFactory_SetConnectAttemptCount</code>	Modify the connection attempts setting.
<code>tibemsConnectionFactory_SetConnectAttemptDelay</code>	Modify the connection delay setting.
<code>tibemsConnectionFactory_SetConnectAttemptTimeout</code>	Modify the connection timeout setting.
<code>tibemsConnectionFactory_SetMetric</code>	Modify the load balancing metric.
<code>tibemsConnectionFactory_SetMulticastDaemon</code>	Set the port on which the client will connect to the multicast daemon.
<code>tibemsConnectionFactory_SetMulticastEnabled</code>	Set whether message consumers subscribed to multicast-enabled topics will receive messages over multicast.
<code>tibemsConnectionFactory_SetPkPassword</code>	Set the SSL private key password for the connection factory.
<code>tibemsConnectionFactory_SetReconnectAttemptCount</code>	Modify the reconnection attempts setting.
<code>tibemsConnectionFactory_SetReconnectAttemptDelay</code>	Modify the reconnection delay setting.
<code>tibemsConnectionFactory_SetReconnectAttemptTimeout</code>	Modify the reconnection timeout setting.
<code>tibemsConnectionFactory_SetServerURL</code>	Set the server URL.
<code>tibemsConnectionFactory_SetSSLParams</code>	Set a connection factory's default SSL parameters.
<code>tibemsConnectionFactory_SetSSLProxy</code>	Set a connection factory's parameters for connecting through an SSL proxy.

Function	Description
<a href="#">tibemsConnectionFactory_SetSSLProxyAuth</a>	Set a connection factory's username and password for connecting through an SSL proxy.
<a href="#">tibemsConnectionFactory_SetUserName</a>	Set a connection factory's username.
<a href="#">tibemsConnectionFactory_SetUserPassword</a>	Set the password used by the connection factory to authenticate itself with the EMS Server.
<a href="#">tibemsUFOConnectionFactory_Create</a>	Create an unshared state connection factory.
<a href="#">tibemsUFOConnectionFactory_CreateFromConnectionFactory</a>	Create an unshared state connection factory from a normal connection factory.
<a href="#">tibemsUFOConnectionFactory_RecoverConnection</a>	Recover an unshared state connection.

### Administered Objects

Administered objects let administrators configure EMS behavior at the enterprise level. Administrators define these objects, and client programs use them. This arrangement relieves program developers and end users of the responsibility for correct configuration.

### See Also

[tibemsLookupContext](#)

## **tibemsConnectionFactory\_Create**

Function

### Purpose

Create a connection factory.

### C Declaration

```
tibemsConnectionFactory tibemsConnectionFactory_Create( void );
```

### COBOL Call

```
CALL "tibemsConnectionFactory_Create"  
      RETURNING factory  
END-CALL.
```



factory has usage pointer.

**Remarks**

The resulting connection factory object is empty. This call does not attempt to access the repository (see also [tibemsLookupContext](#)).

**See Also**

[tibemsLookupContext](#)

**tibemsConnectionFactory\_CreateConnection**

Function

**Purpose**

Create a connection object.

**C Declaration**

```
tibems_status tibemsConnectionFactory_CreateConnection(
    tibemsConnectionFactory factory,
    tibemsConnection* connection,
    const char * username,
    const char * password );
```

**COBOL Call**

```
CALL "tibemsConnectionFactory_CreateConnection"
    USING BY VALUE factory,
         BY REFERENCE connection,
         BY REFERENCE username,
         BY REFERENCE password,
         RETURNING tibems-status
END-CALL.
```



factory and connection have usage pointer.

**Parameters**

Parameter	Description
factory	Use this connection factory to create a connection.
connection	The function stores the new connection object in this location.
userName	The connection object presents this user identity to the server. Set to NULL if the server isn't authenticating or authorizing users.
password	The connection object authenticates the user identity with this password. Set to NULL if the server isn't authenticating or authorizing users.

**Remarks**

When the identity parameters are null, the connection object presents a default user identity. If the server configuration permits that anonymous user, then the call succeeds.

**See Also**

[tibemsConnection](#)

## tibemsConnectionFactory\_CreateXAConnection

Function

### Purpose

Create an XA connection object.

### C Declaration

```
extern tibems_status
tibemsConnectionFactory_CreateXAConnection(
    tibemsConnectionFactory factory,
    tibemsConnection* connection,
    const char* username,
    const char* password);
```

### COBOL Call

```
CALL "tibemsConnectionFactory_CreateXAConnection"
    USING BY VALUE factory,
         BY REFERENCE connection,
         BY REFERENCE username,
         BY REFERENCE password,
         RETURNING tibems-status
END-CALL.
```



factory and connection have usage pointer.

### Parameters

Parameter	Description
factory	Use this connection factory to create a connection.
connection	The function stores the new connection object in this location.
userName	The connection object presents this user identity to the server. Set to NULL if the server isn't authenticating or authorizing users.
password	The connection object authenticates the user identity with this password. Set to NULL if the server isn't authenticating or authorizing users.

### Remarks

When the identity parameters are null, the connection object presents a default user identity. If the server configuration permits that anonymous user, then the call succeeds.

### See Also

[tibemsConnection](#)

## tibemsConnectionFactory\_Destroy

Function

### Purpose

Destroy a connection factory object.

## C Declaration

```
tibems_status tibemsConnectionFactory_Destroy(
    tibemsConnectionFactory factory )
```

## COBOL Call

```
CALL "tibemsConnectionFactory_Destroy"
    USING BY VALUE factory,
         RETURNING tibems-status
END-CALL.
```



factory has usage pointer.

## Parameters

Parameter	Description
factory	Destroy this connection factory.

## Remarks

Destroy an object within the program. It does not affect objects in the repository.

## tibemsConnectionFactory\_GetSSLProxyHost

Function

## Purpose

Get the SSL proxy host from a connection factory.

## C Declaration

```
tibems_status tibemsConnectionFactory_GetSSLProxyHost(
    tibemsConnectionFactory factory,
    const char** proxy_host);
```

## COBOL Call

```
CALL "tibemsConnectionFactory_GetSSLProxyHost"
    USING BY VALUE factory,
         BY REFERENCE proxy-host,
         RETURNING tibems-status
END-CALL.
```



factory and proxy-host have usage pointer.

## Parameters

Parameter	Description
factory	Get the SSL proxy host from this connection factory.
proxy_host	The function stores the proxy host in this location.

## See Also

[tibemsConnectionFactory\\_SetSSLProxy](#)

## tibemsConnectionFactory\_GetSSLProxyPort

Function

### Purpose

Get the SSL proxy port from a connection factory.

### C Declaration

```
tibems_status tibemsConnectionFactory_GetSSLProxyPort(
    tibemsConnectionFactory factory,
    tibems_int* proxy_port);
```

### COBOL Call

```
CALL "tibemsConnectionFactory_GetSSLProxyPort"
    USING BY VALUE factory,
         BY REFERENCE proxy-port,
    RETURNING tibems-status
END-CALL.
```



factory has usage pointer.

### Parameters

Parameter	Description
factory	Get the SSL proxy port number from this connection factory.
proxy_port	The function stores the proxy port in this location.

### See Also

[tibemsConnectionFactory\\_SetSSLProxy](#)

## tibemsConnectionFactory\_GetSSLProxyUser

Function

### Purpose

Get the SSL proxy username from a connection factory.

### C Declaration

```
tibems_status tibemsConnectionFactory_GetSSLProxyUser(
    tibemsConnectionFactory factory,
    const char** proxy_user);
```

### COBOL Call

```
CALL "tibemsConnectionFactory_GetSSLProxyUser"
    USING BY VALUE factory,
         BY REFERENCE proxy-user,
    RETURNING tibems-status
END-CALL.
```



factory and proxy-user have usage pointer.

**Parameters**

Parameter	Description
factory	Get the SSL proxy username from this connection factory.
proxy_user	The function stores the proxy user name in this location.

**See Also**

[tibemsConnectionFactory\\_SetSSLProxyAuth](#)

**tibemsConnectionFactory\_GetSSLProxyPassword**

Function

**Purpose**

Get the SSL proxy password from a connection factory.

**C Declaration**

```
tibems_status tibemsConnectionFactory_GetSSLProxyPassword(
    tibemsConnectionFactory factory,
    const char** proxy_password);
```

**COBOL Call**

```
CALL "tibemsConnectionFactory_GetSSLProxyPassword"
    USING BY VALUE factory,
         BY REFERENCE proxy-password,
    RETURNING tibems-status
END-CALL.
```



factory and proxy-password have usage pointer.

**Parameters**

Parameter	Description
factory	Get the SSL proxy password from this connection factory.
proxy_password	The function stores the proxy password in this location.

**See Also**

[tibemsConnectionFactory\\_SetSSLProxyAuth](#)

**tibemsConnectionFactory\_Print**

Function

**Purpose**

Print the parameters set in a connection factory object.

## C Declaration

```
tibems_status tibemsConnectionFactory_Print(
    tibemsConnectionFactory factory);
```

## COBOL Call

```
CALL "tibemsConnectionFactory_Print"
    USING BY VALUE factory,
         RETURNING tibems-status
END-CALL.
```



factory has usage pointer.

## Parameters

Parameter	Description
factory	Print the parameters set in this connection factory.

## See Also

[tibemsConnectionFactory\\_PrintToBuffer](#)

## tibemsConnectionFactory\_PrintToBuffer

Function

## Purpose

Print the parameters set in a connection factory object to a buffer.

## C Declaration

```
extern tibems_status tibemsConnectionFactory_PrintToBuffer(
    tibemsConnectionFactory factory,
    char* buffer,
    tibems_int maxlen);
```

## COBOL Call

```
CALL "tibemsConnectionFactory_PrintToBuffer"
    USING BY VALUE factory,
         BY REFERENCE buffer,
         BY VALUE maxlen
         RETURNING tibems-status
END-CALL.
```



factory has usage pointer.

## Parameters

Parameter	Description
factory	Print the parameters set in this connection factory.
buffer	Location to store the string representation of the connection factory.
maxlen	The size of the buffer.

**See Also**

[tibemsConnectionFactory\\_Print](#)

**tibemsConnectionFactory\_SetClientID**

Function

**Purpose**

Set the client ID of a connection factory object.

**C Declaration**

```
tibems_status tibemsConnectionFactory_SetClientID(
    tibemsConnectionFactory factory,
    const char* cid );
```

**COBOL Call**

```
CALL "tibemsConnectionFactory_SetClientID"
    USING BY VALUE factory,
         BY REFERENCE cid,
         RETURNING tibems-status
END-CALL.
```



factory has usage pointer.

**Parameters**

Parameter	Description
factory	Set the client ID of this connection factory.
type	Set the ID to this string.

**Remarks**

A client ID string lets the server associate a client-specific factory with each client program. When such a factory already exists, the server supplies that factory to the client. If a factory does not yet exist for the client, the server creates one, and stores it for future use by that specific client.

**tibemsConnectionFactory\_SetConnectAttemptCount**

Function

**Purpose**

Modify the connection attempts setting.

**C Declaration**

```
tibems_status tibemsConnectionFactory_SetConnectAttemptCount(
    tibemsConnectionFactory factory,
    tibems_int connAttempts );
```

**COBOL Call**

```
CALL "tibemsConnectionFactory_SetConnectAttemptCount"
    USING BY VALUE factory,
         BY VALUE connAttempts,
```

```
RETURNING tibems-status
END-CALL.
```



factory has usage pointer.

### Parameters

Parameter	Description
factory	Set the connection attempts parameter of this connection factory.
connAttempts	This value limits the number of times that a connection object attempts to establish a connection to the server. The minimum value is 1.  When this property is not set, the default value is 2.

### See Also

[tibemsConnectionFactory\\_SetConnectAttemptDelay](#)  
[tibemsConnectionFactory\\_SetConnectAttemptTimeout](#)

*Set Connection Attempts, Timeout and Delay Parameters in the TIBCO Enterprise Message Service User's Guide*

## tibemsConnectionFactory\_SetConnectAttemptDelay

Function

### Purpose

Modify the connection delay setting.

### C Declaration

```
tibems_status tibemsConnectionFactory_SetConnectAttemptDelay(
    tibemsConnectionFactory factory,
    tibems_int delay );
```

### COBOL Call

```
CALL "tibemsConnectionFactory_SetConnectAttemptDelay"
    USING BY VALUE factory,
         BY VALUE delay,
         RETURNING tibems-status
END-CALL.
```



factory has usage pointer.

### Parameters

Parameter	Description
factory	Set the connection delay parameter of this connection factory.
delay	This value determines the time (in milliseconds) between connection attempts. The minimum value is 250.  When this property is not set, the default value is 500.

**See Also**

[tibemsConnectionFactory\\_SetConnectAttemptCount](#)  
[tibemsConnectionFactory\\_SetConnectAttemptTimeout](#)

*Set Connection Attempts, Timeout and Delay Parameters in the TIBCO Enterprise Message Service User's Guide*

**tibemsConnectionFactory\_SetConnectAttemptTimeout**

Function

**Purpose**

Modify the connection timeout setting.

**C Declaration**

```
tibems_status tibemsConnectionFactory_SetConnectAttemptTimeout(
    tibemsConnectionFactory factory,
    tibems_int timeout );
```

**COBOL Call**

```
CALL "tibemsConnectionFactory_SetConnectAttemptTimeout"
    USING BY VALUE factory,
         BY VALUE timeout,
         RETURNING tibems-status
END-CALL.
```



factory has usage pointer.

**Parameters**

Parameter	Description
factory	Set the connection timeout parameter of this connection factory.
timeout	<p>This value determines the maximum time (in milliseconds) the client will wait for a connection to the server to be established.</p> <p>The minimum permitted timeout is 100 milliseconds. However, under stress conditions, this minimum may be too low to enable reliable connection creation. In such cases, a value of greater than 1000 is suggested. Zero is a special value, which specifies no timeout.</p> <p>Note that the maximum can be exceeded if the timeout is set to be less than 1000. If the value provided is less than 1000, the creation of the socket is subject to its own minimum of 1 second, which can impact the timeout.</p> <p>When this property is not set, the default value is 0.</p>

**See Also**

[tibemsConnectionFactory\\_SetConnectAttemptCount](#)  
[tibemsConnectionFactory\\_SetConnectAttemptDelay](#)

*Set Connection Attempts, Timeout and Delay Parameters in the TIBCO Enterprise Message Service User's Guide*

## tibemsConnectionFactory\_SetMetric

Function

### Purpose

Modify the load balancing metric.

### C Declaration

```
tibems_status tibemsConnectionFactory_SetMetric(
    tibemsConnectionFactory factory,
    tibemsFactoryLoadBalanceMetric metric );
```

### COBOL Call

```
CALL "tibemsConnectionFactory_SetMetric"
    USING BY VALUE factory,
         BY VALUE metric,
         RETURNING tibems-status
END-CALL.
```



factory has usage pointer.

### Parameters

Parameter	Description
factory	Set the load balancing metric of this connection factory.
metric	Use this metric.

### Remarks

When the connection factory balances the client load among several servers, it uses this metric to determine the least loaded server, so the connection factory can create a connection to it. For values, see [tibemsFactoryLoadBalanceMetric](#).

## tibemsConnectionFactory\_SetMulticastDaemon

Function



Along with the multicast feature, this function was deprecated in software release 8.3.0 and will no longer be supported in future releases.

### Purpose

Set the port on which the client will connect to the multicast daemon.

### C Declaration

```
tibems_status tibemsConnectionFactory_SetMulticastDaemon(
    tibemsConnectionFactory factory,
    const char* multicastDaemon);
```

### Parameters

Parameter	Description
factory	Set the multicast daemon for this connection factory.

Parameter	Description
<code>multicastDaemon</code>	The port number for the multicast daemon that connections created using this factory will connect to.

### Remarks

A connection to the multicast daemon is required when multicast is enabled and a consumer is subscribed to a multicast-enabled topic. Setting the port with this method will override the default port supplied by the server.

This call is not supported in COBOL, and is not supported on z/OS and IBM i systems.

### See Also

[tibemsConnectionFactory\\_SetMulticastEnabled](#)

[tibems\\_SetMulticastDaemon](#)

## tibemsConnectionFactory\_SetMulticastEnabled

Function



Along with the multicast feature, this function was deprecated in software release 8.3.0 and will no longer be supported in future releases.

### Purpose

Set whether message consumers subscribed to multicast-enabled topics will receive messages over multicast.

### C Declaration

```
tibems_status tibemsConnectionFactory_SetMulticastEnabled(
    tibemsConnectionFactory factory,
    tibems_bool multicastEnabled);
```

### Parameters

Parameter	Description
<code>factory</code>	Enable or disable multicast capabilities for connections created by this factory
<code>multicastEnabled</code>	When true, multicast is enabled. When false, multicast is disabled.

### Remarks

When enabled, message consumers using a connection created by this factory, and which are subscribed to a multicast-enabled topic will receive messages over multicast. The default is enabled.

This call is not supported in COBOL, and is not supported on z/OS and IBM i systems.

### See Also

[tibemsConnectionFactory\\_SetMulticastDaemon](#)

[tibems\\_SetMulticastEnabled](#)

## tibemsConnectionFactory\_SetPkPassword

Function

### Purpose

Set the SSL private key password for the connection factory.

### C Declaration

```
extern tibems\_status tibemsConnectionFactory_SetPkPassword(
    tibemsConnectionFactory factory,
    const char* pk_password);
```

### COBOL Call

```
CALL "tibemsConnectionFactory_SetPkPassword"
    USING BY VALUE factory,
          BY REFERENCE pk-password,
          RETURNING tibems-status
END-CALL.
```



factory has usage pointer.

On IBM z/OS systems, the pk-password must always be a null value.

### Parameters

Parameter	Description
factory	Set the SSL password for this connection factory.
pk_password	Connections created by the connection factory decode their SSL private key using this password when establishing SSL communication.

### Remarks

It is an error to call this function on a connection factory for which an [tibemsSSLParams](#) struct is not yet set.

Notice that this SSL private key encryption password is distinct from the server authentication password, and from the proxy authentication password.

### See Also

[tibemsSSLParams](#)

[tibemsConnectionFactory\\_SetSSLParams](#)

## tibemsConnectionFactory\_SetReconnectAttemptCount

Function

### Purpose

Modify the reconnection attempts setting.

### C Declaration

```
tibems\_status tibemsConnectionFactory_SetReconnectAttemptCount(
    tibemsConnectionFactory factory,
    tibems\_int connAttempts );
```

## COBOL Call

```
CALL "tibemsConnectionFactory_SetReconnectAttemptCount"
    USING BY VALUE factory,
          BY VALUE connAttempts,
          RETURNING tibems-status
END-CALL.
```



factory has usage pointer.

### Parameters

Parameter	Description
factory	Set the reconnection attempts parameter of this connection factory.
connAttempts	This value limits the number of times that a connection object attempts to reestablish a connection to the server. The minimum value is 1.  When this property is not set, the default value is 4.

### See Also

[tibemsConnectionFactory\\_SetReconnectAttemptDelay](#)

[tibemsConnectionFactory\\_SetReconnectAttemptTimeout](#)

*Set Reconnection Failure Parameters* in the *TIBCO Enterprise Message Service User's Guide*.

## tibemsConnectionFactory\_SetReconnectAttemptDelay

Function

### Purpose

Modify the reconnection delay setting.

### C Declaration

```
tibems_status tibemsConnectionFactory_SetReconnectAttemptDelay(
    tibemsConnectionFactory factory,
    tibems_int delay );
```

## COBOL Call

```
CALL "tibemsConnectionFactory_SetReconnectAttemptDelay"
    USING BY VALUE factory,
          BY VALUE delay,
          RETURNING tibems-status
END-CALL.
```



factory has usage pointer.

### Parameters

Parameter	Description
factory	Set the reconnection delay parameter of this connection factory.

Parameter	Description
delay	This value determines the time (in milliseconds) between reconnection attempts. The minimum value is 250.  When this property is not set, the default value is 500.

**See Also**

[tibemsConnectionFactory\\_SetReconnectAttemptCount](#)  
[tibemsConnectionFactory\\_SetReconnectAttemptTimeout](#)

*Set Reconnection Failure Parameters* in the *TIBCO Enterprise Message Service User's Guide*.

**tibemsConnectionFactory\_SetReconnectAttemptTimeout**

Function

**Purpose**

Modify the reconnection timeout setting.

**C Declaration**

```
tibems_status tibemsConnectionFactory_SetReconnectAttemptTimeout(
    tibemsConnectionFactory factory,
    tibems_int timeout );
```

**COBOL Call**

```
CALL "tibemsConnectionFactory_SetReconnectAttemptTimeout"
    USING BY VALUE factory,
         BY VALUE timeout,
         RETURNING tibems-status
END-CALL.
```



factory has usage pointer.

**Parameters**

Parameter	Description
factory	Set the reconnection timeout parameter of this connection factory.
timeout	This value determines the maximum time (in milliseconds) a client will wait for the reconnection to be established. Zero is a special value, which specifies no timeout.  When this property is not set, the default value is 0.

**See Also**

[tibemsConnectionFactory\\_SetReconnectAttemptCount](#)  
[tibemsConnectionFactory\\_SetReconnectAttemptDelay](#)

*Set Reconnection Failure Parameters* in the *TIBCO Enterprise Message Service User's Guide*.

## tibemsConnectionFactory\_SetServerURL

Function

### Purpose

Set the server URL.

### C Declaration

```
tibems_status tibemsConnectionFactory_SetServerURL(
    tibemsConnectionFactory factory,
    const char* url );
```

### COBOL Call

```
CALL "tibemsConnectionFactory_SetServerURL"
    USING BY VALUE factory,
         BY REFERENCE url,
         RETURNING tibems-status
END-CALL.
```



factory has usage pointer.

### Parameters

Parameter	Description
factory	Set the server URL of this connection factory.
url	The factory object contacts the EMS server at this URL, to access a corresponding factory defined by the administrator.

### Reconnect and Fault Tolerance

To enable reconnection behavior and fault tolerance, the connection factory's server URL parameter must be a comma-separated list of two or more URLs. To enable client reconnection in a situation with only one server, you may supply two copies of that server's URL (for example, `tcp://localhost:7222,tcp://localhost:7222`).

Note that `tibemsConnectionFactory_SetServerURL` can be used to set the server URL for a connection factory only once. If the URL has previously been set for the connection factory, `tibemsConnectionFactory_SetServerURL` returns the status code `TIBEMS_EXCEPTION`.

When specifying an IPv6 address, use square brackets around the address specification. For example, `tcp://[2001:cafe::107]:7222`.

## tibemsConnectionFactory\_SetSSLParams

Function

### Purpose

Set a connection factory's default SSL parameters.

### C Declaration

```
tibems_status tibemsConnectionFactory_SetSSLParams(
    tibemsConnectionFactory factory,
    tibemsSSLParams sslparams );
```

## COBOL Call

```
CALL "tibemsConnectionFactory_SetSSLParams"
    USING BY VALUE factory,
          BY VALUE sslparams,
          RETURNING tibems-status
END-CALL.
```



factory and sslparams have usage pointer.

## Parameters

Parameter	Description
factory	Set the default SSL parameters of this connection factory.
sslParams	The connection establishes SSL communication using these parameters.

## See Also

[tibemsSSLParams](#)

[tibemsConnectionFactory\\_CreateConnection](#)

## tibemsConnectionFactory\_SetSSLProxy

Function

### Purpose

Set a connection factory's parameters for connecting through an SSL proxy.

### C Declaration

```
tibems_status tibemsConnectionFactory_SetSSLProxy(
    tibemsConnectionFactory factory,
    const char* proxy_host,
    tibems_int proxy_port);
```

## COBOL Call

```
CALL "tibemsConnectionFactory_SetSSLProxy"
    USING BY VALUE factory,
          BY REFERENCE proxy-host,
          BY VALUE proxy-port,
          RETURNING tibems-status
END-CALL.
```



factory has usage pointer.

## Parameters

Parameter	Description
factory	Set the SSL proxy host and port on this connection factory.
proxy_host	The connection factory establishes SSL communication through a web proxy at this host. Supply a simple hostname, a fully qualified hostname with domain name, or an IP address (dot notation).

Parameter	Description
proxy_port	The connection factory establishes SSL communication through a web proxy on this port.

### Remarks

An SSL proxy lets an EMS application create an SSL connection to an EMS server, even though a firewall separates the application from the server. The proxy usually runs within the firewall's DMZ.

A connection factory contacts the SSL proxy, requesting an SSL connection to the server. The proxy authenticates the application program, and mediates the initial SSL negotiation between application and server. After the SSL connection is established, the application and server use it to communicate directly with one another.

### See Also

[tibemsConnectionFactory\\_GetSSLProxyHost](#)

[tibemsConnectionFactory\\_GetSSLProxyPort](#)

[tibemsConnectionFactory\\_SetSSLProxyAuth](#)

## tibemsConnectionFactory\_SetSSLProxyAuth

Function

### Purpose

Set a connection factory's username and password for connecting through an SSL proxy.

### C Declaration

```
tibems_status
tibemsConnectionFactory_SetSSLProxyAuth(
    tibemsConnectionFactory factory,
    const char* proxy_user,
    const char* proxy_password);
```

### COBOL Call

```
CALL "tibemsConnectionFactory_SetSSLProxy"
    USING BY VALUE factory,
          BY REFERENCE proxy-user,
          BY REFERENCE proxy-password,
          RETURNING tibems-status
END-CALL.
```



factory has usage pointer.

### Parameters

Parameter	Description
factory	Set the username and password on this connection factory.
proxy_user	The connection factory authenticates itself to the SSL proxy using this username.

Parameter	Description
proxy_password	The connection factory authenticates itself to the SSL proxy using this password.

### Remarks

When a connection factory establishes an EMS server connection through an SSL proxy host, the proxy might first require authentication before facilitating a connection. When required, use this call to set that authentication data on the connection factory. Notice that this proxy authentication data is distinct from the server authentication data, and from the SSL private key encryption password.

### See Also

[tibemsConnectionFactory\\_GetSSLProxyUser](#)

[tibemsConnectionFactory\\_GetSSLProxyPassword](#)

[tibemsConnectionFactory\\_SetSSLProxy](#)

## tibemsConnectionFactory\_SetUserName

Function

### Purpose

Set a connection factory's username.

### C Declaration

```
tibems_status
tibemsConnectionFactory_SetUserName(
    tibemsConnectionFactory factory,
    const char* username);
```

### COBOL Call

```
CALL "tibemsConnectionFactory_SetUserName"
    USING BY VALUE factory,
         BY REFERENCE username,
         RETURNING tibems-status
END-CALL.
```



factory has usage pointer.

### Parameters

Parameter	Description
factory	Set the username on this connection factory.
username	The connection factory identifies itself using this username.

### Remarks

When a connection factory establishes an EMS server connection, the EMS server requests identification. Use this call to set the username that the connection factory uses to identify itself to the EMS server. Notice that this server authentication data is different from SSL authentication data.

**See Also**

[tibemsConnectionFactory\\_SetUserPassword](#)

**tibemsConnectionFactory\_SetUserPassword**

Function

**Purpose**

Set the password used by the connection factory to authenticate itself with the EMS Server.

**C Declaration**

```
tibems_status
tibemsConnectionFactory_SetUserPassword(
    tibemsConnectionFactory factory,
    const char* password);
```

**COBOL Call**

```
CALL "tibemsConnectionFactory_SetUserPassword"
    USING BY VALUE factory,
         BY REFERENCE password,
         RETURNING tibems-status
END-CALL.
```



factory has usage pointer.

**Parameters**

Parameter	Description
factory	Set the password on this connection factory.
password	The connection factory authenticates itself using this password.

**Remarks**

When a connection factory establishes an EMS server connection, the EMS server requests authentication. Use this call to set the password that the connection factory uses to authenticate itself with the EMS server. Notice that this server authentication data is different from SSL authentication data.

**See Also**

[tibemsConnectionFactory\\_SetUserName](#)

**tibemsUFOConnectionFactory\_Create**

Function

**Purpose**

Create an unshared state connection factory.

**C Declaration**

```
tibemsConnectionFactory tibemsUFOConnectionFactory_Create(void);
```

**COBOL Call**

```
CALL "tibemsUFOConnectionFactory_Create"
    RETURNING factory
END-CALL.
```



factory has usage pointer.

**Remarks**

The resulting unshared state connection factory object is empty. This call does not attempt to access the repository (see also [tibemsLookupContext](#)).

**See Also**

[tibemsLookupContext](#)

**tibemsUFOConnectionFactory\_CreateFromConnectionFactory**

Function

**Purpose**

Create an unshared state connection factory from a normal connection factory.

**C Declaration**

```
tibemsConnectionFactory
tibemsUFOConnectionFactory_CreateFromConnectionFactory(
    tibemsConnectionFactory emsFactory);
```

**COBOL Call**

```
CALL "tibemsUFOConnectionFactory_CreateFromConnectionFactory"
    USING BY VALUE emsFactory
    RETURNING factory
END-CALL.
```



factory has usage pointer.

**Parameters**

Parameter	Description
emsFactory	Connections created by this unshared state connection factory connect to a server in the URL list of this connection factory.

**See Also**

[tibemsUFOConnectionFactory\\_Create](#)

**tibemsUFOConnectionFactory\_RecoverConnection**

Function

**Purpose**

Recover an unshared state connection.

## C Declaration

```
tibems_status
tibemsUFOConnectionFactory_RecoverConnection(
    tibemsConnectionFactory ufoFactory,
    tibemsConnection ufoConnection);
```

## COBOL Call

```
CALL "tibemsUFOConnectionFactory_RecoverConnection"
    USING BY VALUE ufoFactory,
         BY VALUE ufoConnection,
         RETURNING tibems-status
END-CALL.
```



ufoConnection has usage pointer.

## Parameters

Parameter	Description
ufoFactory	The unshared state connection factory used to create the unshared state connection.
ufoConnection	The unshared state connection to be recovered.

## Remarks

This function is used to recover the broken connection on another available server.

## See Also

*Unshared State Failover Process* in the *TIBCO Enterprise Message Service User's Guide*.

## tibemsFactoryLoadBalanceMetric

*Type*

### Purpose

Define enumerated load balancing constants.

### Remarks

When a connection factory balances the client load among several servers, it uses this metric to determine the least loaded server, so the connection factory can create a connection to it.

Constant	Description
TIBEMS_FACTORY_LOAD_BALANCE_METRIC_NONE	Indicate absence of any load balancing metric.
TIBEMS_FACTORY_LOAD_BALANCE_METRIC_CONNECTIONS	The connection factory balances the connection load among several servers by creating a connection to the server with the fewest number of connections.

Constant	Description
TIBEMS_FACTORY_LOAD_BALANCE_METRIC_BYTE_RATE	The connection factory balances the connection load among several servers by creating a connection to the server with the lowest total byte rate (input and output).

## COBOL

```
* NOTE: LBM is an acronym for LOAD-BALANCE-METRIC
01 TIBEMS-FACTORY-LBM-TYPES.
   05 tibemsFactoryLoadBalanceMetric    PIC S9(8) BINARY.
      88 TIBEMS-FACTORY-LBM-NONE        VALUE 0.
      88 TIBEMS-FACTORY-LBM-CONNECTIONS VALUE 1.
      88 TIBEMS-FACTORY-LBM-BYTE-RATE   VALUE 2.
```

## See Also

[tibemsConnectionFactory](#)

[tibemsConnectionFactory\\_SetMetric](#)

# Session

---

A session is a single-threaded context for producing and consuming messages.

## **tibemsSession**

*Type*

### **Purpose**

Organizes context for message activity.

### **Remarks**

Sessions combine several roles:

- Create message producers and consumers
- Create message objects
- Create temporary destinations
- Create dynamic destinations
- Create queue browsers
- Serialize for inbound and outbound messages
- Serialize for asynchronous message events (or message listeners) of its consumer objects
- Cache inbound messages (until the program acknowledges them).
- Transaction support (when enabled).

### **Single Thread**

The JMS specification restricts programs to use each session within a single thread.

### **Associated Objects**

The same single-thread restriction applies to objects associated with a session—namely, messages, message consumers, durable subscribers, message producers, queue browsers, and temporary destinations (however, static and dynamic destinations are exempt from this restriction).

### **Corollary**

One consequence of this rule is that all the consumers of a session must deliver messages in the same mode—either synchronously or asynchronously.

### **Asynchronous**

In asynchronous delivery, the program registers message handler events or message listeners with the session's consumer objects. An internal dispatcher thread delivers messages to those event handlers or listeners (in all the session's consumer objects). No other thread may use the session (nor objects created by the session).

### **Synchronous**

In synchronous delivery, the program explicitly begins a thread for the session. That thread processes inbound messages and produces outbound messages, serializing this activity among the session's producers and consumers. Functions that request the next message (such as [tibemsMsgConsumer\\_Receive](#)) can organize the thread's activity.

### **Close**

The only exception to the rule restricting session calls to a single thread is the function [tibemsSession\\_Close](#); programs can call Close from any thread at any time.

## Transactions

A session has either transaction or non-transaction semantics. When a program specifies transaction semantics, the session object cooperates with the server, and all messages that flow through the session become part of a transaction.

- When the program calls `tibemsSession_Commit`, the session acknowledges all inbound messages in the current transaction, and the server delivers all outbound messages in the current transaction to their destinations.
- If the program calls `tibemsSession_Rollback`, the session recovers all inbound messages in the current transaction (so the program can consume them in a new transaction), and the server destroys all outbound messages in the current transaction.

After these actions, both Commit and Rollback immediately begin a new transaction.

Function	Description
<b>Messages</b>	
<code>tibemsSession_CreateBytesMessage</code>	Create a byte array message.
<code>tibemsSession_CreateMapMessage</code>	Create a map message.
<code>tibemsSession_CreateMessage</code>	Create a message.
<code>tibemsSession_CreateStreamMessage</code>	Create a stream message.
<code>tibemsSession_CreateTextMessage</code>	Create a text message.
<b>Destinations</b>	
<code>tibemsSession_CreateBrowser</code>	Create a queue browser.
<code>tibemsSession_CreateTemporaryQueue</code>	Create a temporary queue.
<code>tibemsSession_CreateTemporaryTopic</code>	Create a temporary topic.
<code>tibemsSession_DeleteTemporaryQueue</code>	Delete a temporary queue.
<code>tibemsSession_DeleteTemporaryTopic</code>	Delete a temporary topic.
<b>Consumers &amp; Producers</b>	
<code>tibemsSession_CreateConsumer</code>	Create a message consumer.
<code>tibemsSession_CreateDurableSubscriber</code>	Create a durable topic subscriber.
<code>tibemsSession_CreateProducer</code>	Create a message producer.
<code>tibemsSession_CreateSharedConsumer</code>	Create a shared consumer.
<code>tibemsSession_CreateSharedDurableConsumer</code>	Create a shared durable consumer.

Function	Description
<code>tibemsSession_Unsubscribe</code>	Unsubscribe a durable topic subscription.
<b>Transactions</b>	
<code>tibemsSession_Commit</code>	Commit the open transaction.
<code>tibemsSession_Rollback</code>	Roll back messages in the current transaction.
<code>tibemsSession_GetTransacted</code>	Get the transactional semantics property of a session.
<b>Other</b>	
<code>tibemsSession_Close</code>	Close a session; reclaim resources.
<code>tibemsSession_Recover</code>	Recover from undetermined state during message processing.
<code>tibemsSession_GetAcknowledgeMode</code>	Get the acknowledge mode of a session.

## **tibemsSession\_Close**

*Function*

### **Purpose**

Close a session; reclaim resources.

### **C Declaration**

```
tibems_status tibemsSession_Close(
    tibemsSession session );
```

### **COBOL Call**

```
CALL "tibemsSession_Close"
    USING BY VALUE session,
    RETURNING tibems-status
END-CALL.
```



`session` has usage pointer.

### **Remarks**

Closing a session automatically closes its consumers (except for durable subscribers), producers and browsers.

### **Blocking**

If any message listener or receive call associated with the session is processing a message when the program calls this function, all facilities of the connection and its sessions remain available to those listeners until they return. In the meantime, this function blocks until that processing completes—that is, until all message listeners and receive calls have returned.

## Transactions

Closing a session rolls back the open transaction in the session.

### tibemsSession\_Commit

*Function*

#### Purpose

Commit the open transaction.

#### C Declaration

```
tibems_status tibemsSession_Commit(
    tibemsSession session );
```

#### COBOL Call

```
CALL "tibemsSession_Commit"
    USING BY VALUE session,
    RETURNING tibems-status
END-CALL.
```



session has usage pointer.

#### Remarks

A session (with transaction semantics) always has exactly one open transaction. Message operations associated with the session become part of that transaction. This call commits all the messages within the transaction, and releases any locks. Then it opens a new transaction.

Status Code	Description
TIBEMS_OK	Successful commit.
TIBEMS_ILLEGAL_STATE	The session does not have transaction semantics.
TIBEMS_SECURITY_EXCEPTION	The client lacks permission to send one of messages in the transaction.
TIBEMS_TRANSACTION_FAILED	Commit failed and the server automatically rolled back the transaction.
TIBEMS_TRANSACTION_ROLLBACK	

### tibemsSession\_CreateBrowser

*Function*

#### Purpose

Create a queue browser.

#### C Declaration

```
tibems_status tibemsSession_CreateBrowser(
    tibemsSession session,
    tibemsQueueBrowser* browser,
    tibemsQueue queue,
    const char* messageSelector );
```

## COBOL Call

```
CALL "tibemsSession_CreateBrowser"
    USING BY VALUE session,
          BY REFERENCE browser,
          BY VALUE queue,
          BY REFERENCE messageSelector,
          RETURNING tibems-status
END-CALL.
```



session, browser and queue have usage pointer.

## Parameters

Parameter	Description
session	Create a browser in this session.
browser	The function stores the new browser object in this location.
queue	Browse this queue.
messageSelector	When non-null, the browser presents only messages that match this selector; see <a href="#">Message Selectors</a> . When null, or the empty string, the browser views all messages in the queue.

## See Also

[tibemsQueue](#)

[tibemsQueueBrowser](#)

## tibemsSession\_CreateBytesMessage

*Function*

### Purpose

Create a byte array message.

### C Declaration

```
tibems_status tibemsSession_CreateBytesMessage(
    tibemsSession session,
    tibemsBytesMsg* bytesMsg );
```

## COBOL Call

```
CALL "tibemsSession_CreateBytesMessage"
    USING BY VALUE session,
          BY REFERENCE bytesMsg,
          RETURNING tibems-status
END-CALL.
```



session and bytesMsg have usage pointer.

## Parameters

Parameter	Description
session	Create the message in this session.
bytesMsg	The function stores the new message object in this location.

## See Also

[tibemsBytesMsg](#)

## tibemsSession\_CreateConsumer

*Function*

### Purpose

Create a message consumer.

### C Declaration

```
tibems_status tibemsSession_CreateConsumer(
    tibemsSession session,
    tibemsMsgConsumer* consumer,
    tibemsDestination destination,
    const char* messageSelector,
    tibems_bool noLocal );
```

### COBOL Call

```
CALL "tibemsSession_CreateConsumer"
    USING BY VALUE session,
         BY REFERENCE consumer,
         BY VALUE destination,
         BY REFERENCE messageSelector,
         BY VALUE noLocal,
         RETURNING tibems-status
END-CALL.
```



session, consumer and destination have usage pointer.

## Parameters

Parameter	Description
session	Create the consumer in this session.
consumer	The function stores the new consumer object in this location.
destination	Create a consumer for this destination. The argument may be any destination (queue or topic).
messageSelector	When non-null, the server filters messages using this selector, so the consumer receives only matching messages; see <a href="#">Message Selectors</a> . When null, or the empty string, the consumer receives messages without filtering.

Parameter	Description
noLocal	When true, the server filters messages so the consumer does not receive messages that originate locally—that is, messages sent through the same connection.  When false, the consumer receives messages with local origin.

### See Also

[tibemsDestination](#)

[tibemsMsgConsumer](#)

## tibemsSession\_CreateDurableSubscriber

*Function*

### Purpose

Create a durable topic subscriber.

### C Declaration

```
tibems_status tibemsSession_CreateDurableSubscriber(
    tibemsSession session,
    tibemsMsgConsumer* msgConsumer,
    tibemsTopic topic,
    const char* name,
    const char* messageSelector,
    tibems_bool noLocal );
```

### COBOL Call

```
CALL "tibemsSession_CreateDurableSubscriber"
    USING BY VALUE session,
         BY REFERENCE msgConsumer,
         BY VALUE topic,
         BY REFERENCE name,
         BY REFERENCE messageSelector,
         BY VALUE noLocal,
         RETURNING tibems-status
END-CALL.
```



session, msgConsumer, and topic have usage pointer.

### Parameters

Parameter	Description
session	Create the topic subscriber in this session.
msgConsumer	Store the new message consumer object in this location. Note that the message consumer must be a topic subscriber.
topic	Create a durable subscriber for this topic (which <i>cannot</i> be a <a href="#">tibemsTemporaryTopic</a> ).
name	Unique name that the server uses to associate the subscriber with a subscription.

Parameter	Description
<code>messageSelector</code>	<p>When non-null, the server filters messages using this selector, so the subscriber receives only matching messages; see <a href="#">Message Selectors</a>.</p> <p>When null, or the empty string, the subscriber receives messages without filtering.</p>
<code>noLocal</code>	<p>When true, the server filters messages so the subscriber does not receive messages that originate locally – that is, messages sent through the same connection.</p> <p>When false, the consumer receives messages with local origin.</p>

### Remarks

The server associates a durable subscription with at most one subscriber object at a time. When a subscriber object exists, the subscription is *active*, and the server delivers messages to it; when no subscriber object exists, the subscription is *inactive*.

Durable subscriptions guarantee message delivery across periods during which the subscriber is inactive. The server retains unacknowledged messages until the subscriber acknowledges them, or until the messages expire.

### Subscription Continuity

Continuity across inactive periods uses two data items from the client:

- **Subscription Name**- A parameter of this call.
- **Client ID** - an optional property of the `tibemsConnection` (used only when supplied)

The server uses one or both of these two items to match a subscriber object with its subscription. If a matching subscription exists, and it is inactive, then the server associates it with the subscriber (and the subscription becomes active). The server delivers unacknowledged messages to the subscriber.

If a matching subscription exists, but it is already active, this function fails with `TIBEMS_INVALID_CONSUMER`.

If a matching subscription to the topic does not yet exist, the server creates one.

### Matching Client ID

- If the `tibemsConnection`'s client ID is non-null when a session creates a durable subscription, then only sessions of a connection with the same client ID can attach to that subscription.
- If the `tibemsConnection`'s client ID is null when a session creates a durable subscription, then any session can attach to that subscription (to receive its messages).

### Changing Topic or Selector

Notice that the server does *not* use the topic and message selector arguments to match a subscriber to an existing subscription. As a result, client programs can *change* a subscription by altering either or both of these arguments. The effect is equivalent to deleting the existing subscription (from the server) and creating a new one (albeit with the same client ID and subscription name).

### See Also

[tibemsTopic](#)

[tibemsMsgConsumer](#)

[tibemsConnection](#)

## tibemsSession\_CreateMapMessage

*Function*

### Purpose

Create a map message.

### C Declaration

```
tibems_status tibemsSession_CreateMapMessage(
    tibemsSession session,
    tibemsMapMsg* mapMsg );
```

### COBOL Call

```
CALL "tibemsSession_CreateMapMessage"
    USING BY VALUE session,
         BY REFERENCE mapMsg,
         RETURNING tibems-status
END-CALL.
```



session and mapMsg have usage pointer.

### Parameters

Parameter	Description
session	Create the message in this session.
mapMsg	The function stores the new message object in this location.

### Remarks

The JMS specification requires this call. It is equivalent to [tibemsMapMsg\\_Create](#).

### See Also

[tibemsMapMsg](#)

## tibemsSession\_CreateMessage

*Function*

### Purpose

Create a message.

### C Declaration

```
tibems_status tibemsSession_CreateMessage(
    tibemsSession session,
    tibemsMsg* message );
```

### COBOL Call

```
CALL "tibemsSession_CreateMessage"
    USING BY VALUE session,
         BY REFERENCE message,
         RETURNING tibems-status
END-CALL.
```



session and message have usage pointer.

### Parameters

Parameter	Description
session	Create the message in this session.
message	The function stores the new message object in this location.

### Remarks

The JMS specification requires this call. It is equivalent to [tibemsMsg\\_Create](#).

### See Also

[tibemsMsg](#)

## tibemsSession\_CreateProducer

*Function*

### Purpose

Create a message producer.

### C Declaration

```
tibems_status tibemsSession_CreateProducer(
    tibemsSession session,
    tibemsMsgProducer* producer,
    tibemsDestination destination );
```

### COBOL Call

```
CALL "tibemsSession_CreateProducer"
    USING BY VALUE session,
          BY REFERENCE producer,
          BY VALUE destination,
          RETURNING tibems-status
END-CALL.
```



session, producer and destination have usage pointer.

### Parameters

Parameter	Description
session	Create the producer in this session.
producer	The function stores the new producer object in this location.
destination	When non-null, the producer sends messages to this destination. When null, the client program must specify the destination for each message individually.

**See Also**[tibemsDestination](#)[tibemsMsgProducer](#)**tibemsSession\_CreateSharedConsumer***Function***Purpose**

Create a shared consumer.

**C Declaration**

```

tibems_status tibemsSession_CreateSharedConsumer(
    tibemsSession session,
    tibemsMsgConsumer* consumer,
    tibemsTopic topic,
    const char* sharedSubscriptionName,
    const char* messageSelector );

```

**COBOL Call**

```

CALL "tibemsSession_CreateSharedConsumer"
    USING BY VALUE session,
         BY REFERENCE consumer,
         BY VALUE topic,
         BY REFERENCE sharedSubscriptionName,
         BY REFERENCE messageSelector,
         RETURNING tibems-status
END-CALL.

```



session, consumer and topic have usage pointer.

**Parameters**

Parameter	Description
session	Create the shared non-durable consumer in this session.
consumer	The function stores the new message consumer object in this location.
topic	Create the shared consumer for this topic.
sharedSubscriptionName	The name used to identify the shared non-durable subscription.
messageSelector	When non-null, the server filters messages using this selector, so the consumer receives only matching messages; see <a href="#">Message Selectors</a> .  When null, or the empty string, the consumer receives messages without filtering.

## Remarks

Creates a shared non-durable subscription with the specified name on the specified topic (if one does not already exist), optionally specifying a message selector, and creates a consumer on that subscription.

If a shared non-durable subscription already exists with the same name and client identifier (if set), and the same topic and message selector has been specified, then this method creates a `tibemsMsgConsumer` on the existing subscription.

A non-durable shared subscription is used by a client that needs to be able to share the work of receiving messages from a topic subscription amongst multiple consumers. A non-durable shared subscription may therefore have more than one consumer. Each message from the subscription will be delivered to only one of the consumers on that subscription. Such a subscription is not persisted and is deleted (together with any undelivered messages associated with it) when there are no consumers on it. The term *consumer* here means a `tibemsMsgConsumer` object in any client.

A shared non-durable subscription is identified by a name specified by the client and by the client identifier (which may be unset). An application which subsequently wishes to create a consumer on that shared non-durable subscription must use the same client identifier.

If a shared non-durable subscription already exists with the same name and client identifier (if set) but a different topic or message selector has been specified, and there is a consumer already active (i.e. not closed) on the subscription, then an error will be returned.

There is no restriction on durable subscriptions and shared non-durable subscriptions having the same name and `clientId` (which may be unset). Such subscriptions would be completely separate.

## See Also

[tibemsSession\\_CreateSharedDurableConsumer](#)

## **tibemsSession\_CreateSharedDurableConsumer**

*Function*

### Purpose

Create a shared durable consumer.

### C Declaration

```
tibems_status tibemsSession_CreateSharedDurableConsumer(
    tibemsSession session,
    tibemsMsgConsumer* consumer,
    tibemsTopic topic,
    const char* durableName,
    const char* messageSelector );
```

### COBOL Call

```
CALL "tibemsSession_CreateSharedDurableConsumer"
    USING BY VALUE session,
         BY REFERENCE consumer,
         BY VALUE topic,
         BY REFERENCE durableName,
         BY REFERENCE messageSelector,
         RETURNING tibems-status
END-CALL.
```



session, consumer and topic have usage pointer.

## Parameters

Parameter	Description
<code>session</code>	Create the shared durable consumer in this session.
<code>consumer</code>	The function stores the new message consumer object in this location.
<code>topic</code>	Create the shared durable consumer for this topic.
<code>durableName</code>	The name used to identify the shared durable subscription.
<code>messageSelector</code>	When non-null, the server filters messages using this selector, so the consumer receives only matching messages; see <a href="#">Message Selectors</a> . When null, or the empty string, the consumer receives messages without filtering.

## Remarks

Creates a shared durable subscription on the specified topic (if one does not already exist), optionally specifying a message selector, and creates a consumer on that durable subscription.

A durable subscription is used by an application that needs to receive all the messages published on a topic, including the ones published when there is no active consumer associated with it. The server retains a record of this durable subscription and ensures that all messages from the topic's publishers are retained until they are delivered to, and acknowledged by, a consumer on this durable subscription, or until they have expired.

A durable subscription will continue to accumulate messages until it is deleted using the `tibemsSession_Unsubscribe` function.

This method may only be used with shared durable subscriptions. Any durable subscription created using this method is shared. This means that multiple active (that is, not closed) consumers on the subscription may exist at the same time. The term *consumer* here means a `tibemsMsgConsumer` object in any client.

A shared durable subscription is identified by a name specified by the client and by the client identifier (which may be unset). An application which subsequently wishes to create a consumer on that shared durable subscription must use the same client identifier.

If a shared durable subscription already exists with the same name and client identifier (if set), and the same topic and message selector have been specified, then this method creates a `tibemsMsgConsumer` on the existing shared durable subscription.

If a shared durable subscription already exists with the same name and client identifier (if set), but a different topic or message selector has been specified, and there is no consumer already active (that is, not closed) on the durable subscription, then this is equivalent to unsubscribing (deleting) the old one and creating a new one.

If a shared durable subscription already exists with the same name and client identifier (if set) but a different topic or message selector has been specified, and there is a consumer already active (that is, not closed) on the durable subscription, then an error is returned.

A shared durable subscription and an unshared durable subscription may not have the same name and client identifier (if set). If an unshared durable subscription already exists with the same name and client identifier (if set) then an error is returned.

There is no restriction on durable subscriptions and shared non-durable subscriptions having the same name and `clientId` (which may be unset). Such subscriptions would be completely separate.

**See Also**[tibemsSession\\_CreateSharedConsumer](#)[tibemsSession\\_Unsubscribe](#)**tibemsSession\_CreateStreamMessage***Function***Purpose**

Create a stream message.

**C Declaration**

```
tibems_status tibemsSession_CreateStreamMessage(
    tibemsSession session,
    tibemsStreamMsg* streamMsg );
```

**COBOL Call**

```
CALL "tibemsSession_CreateStreamMessage"
    USING BY VALUE session,
         BY REFERENCE streamMsg,
         RETURNING tibems-status
END-CALL.
```



session and streamMsg have usage pointer.

**Parameters**

Parameter	Description
session	Create the message in this session.
streamMsg	The function stores the new message object in this location.

**Remarks**

The JMS specification requires this call. It is equivalent to [tibemsStreamMsg\\_Create](#).

**See Also**[tibemsStreamMsg](#)**tibemsSession\_CreateTemporaryQueue***Function***Purpose**

Create a temporary queue.

**C Declaration**

```
tibems_status tibemsSession_CreateTemporaryQueue(
    tibemsSession session,
    tibemsTemporaryQueue* tmpQueue );
```

## COBOL Call

```
CALL "tibemsSession_CreateTemporaryQueue"
    USING BY VALUE session,
          BY REFERENCE tmpQueue,
          RETURNING tibems-status
END-CALL.
```



session and tmpQueue have usage pointer.

## Parameters

Parameter	Description
session	Create the queue in this session.
tmpQueue	The function stores the new temporary queue object in this location.

## Remarks

A temporary queue lasts no longer than the connection. That is, when the connection is closed or broken, the server deletes temporary queues associated with the connection.

If the named queue already exists at the server, then this function returns that queue. (That queue can be either static or dynamic.)

If the named queue does not yet exist at the server, and the server allows dynamic queue, then this function creates a dynamic queue.

Dynamic destinations are provider-specific, so programs that use them might not be portable to other providers.

## See Also

[tibemsTemporaryQueue](#)

## tibemsSession\_CreateTemporaryTopic

*Function*

### Purpose

Create a temporary topic.

### C Declaration

```
tibems_status tibemsSession_CreateTemporaryTopic(
    tibemsSession session,
    tibemsTemporaryTopic* tmpTopic );
```

## COBOL Call

```
CALL "tibemsSession_CreateTemporaryTopic"
    USING BY VALUE session,
          BY REFERENCE tmpTopic,
          RETURNING tibems-status
END-CALL.
```



session and tmpTopic have usage pointer.

## Parameters

Parameter	Description
session	Create the topic in this session.
tmpTopic	The function stores the new temporary topic object in this location.

## Remarks

A temporary topic lasts no longer than the connection. That is, when the connection is closed or broken, the server deletes temporary topic associated with the connection.

If the named topic already exists at the server, then this function returns that topic. (That topic can be either static or dynamic.)

If the named topic does not yet exist at the server, and the server allows dynamic topics, then this function creates a dynamic topic.

Dynamic destinations are provider-specific, so programs that use them might not be portable to other providers.

## See Also

[tibemsTemporaryTopic](#)

## tibemsSession\_CreateTextMessage

*Function*

### Purpose

Create a text message.

### C Declaration

```
tibems_status tibemsSession_CreateTextMessage(
    tibemsSession session,
    tibemsTextMsg* textMsg );

tibems_status tibemsSession_CreateTextMessageEx(
    tibemsSession session,
    tibemsTextMsg* textMsg
    const char* text );
```

### COBOL Call

```
CALL "tibemsSession_CreateTextMessage"
    USING BY VALUE session,
         BY REFERENCE textMsg,
         RETURNING tibems-status
END-CALL.

CALL "tibemsSession_CreateTextMessageEx"
    USING BY VALUE session,
         BY REFERENCE textMsg,
         BY REFERENCE text,
         RETURNING tibems-status
END-CALL.
```



session and textMsg have usage pointer.

## Parameters

Parameter	Description
session	Create the message in this session.
textMsg	Store the new message object in this location.
text	Create a text message with this text as its body.

## Remarks

The JMS specification requires these calls. It is equivalent to `tibemsTextMsg_Create`.

## See Also

[tibemsTextMsg](#)

## tibemsSession\_DeleteTemporaryQueue

*Function*

### Purpose

Delete a temporary queue.

### C Declaration

```
tibems_status tibemsSession_DeleteTemporaryQueue(
    tibemsSession session,
    tibemsTemporaryQueue tmpQueue );
```

### COBOL Call

```
CALL "tibemsSession_DeleteTemporaryQueue"
    USING BY VALUE session,
         BY VALUE tmpQueue,
         RETURNING tibems-status
END-CALL.
```



session and tmpQueue have usage pointer.

## Parameters

Parameter	Description
session	Delete a temporary queue from this session.
tmpQueue	Delete this temporary queue.

## Remarks

When a client deletes a temporary queue, the server deletes any unconsumed messages in the queue.

If the client still has listeners or receivers for the queue, then this delete call returns `TIBEMS_ILLEGAL_STATE`.

**See Also**

[tibemsTemporaryQueue](#)

**tibemsSession\_DeleteTemporaryTopic**

*Function*

**Purpose**

Delete a temporary topic.

**C Declaration**

```
tibems_status tibemsSession_DeleteTemporaryTopic(
    tibemsSession session,
    tibemsTemporaryTopic tmpTopic );
```

**COBOL Call**

```
CALL "tibemsSession_DeleteTemporaryTopic"
    USING BY VALUE session,
         BY VALUE tmpTopic,
         RETURNING tibems-status
END-CALL.
```



session and tmpTopic have usage pointer.

**Parameters**

Parameter	Description
session	Delete a temporary topic from this session.
tmpTopic	Delete this temporary topic.

**Remarks**

When a client deletes a temporary topic, the server deletes any unconsumed messages in the topic.

If the client still has listeners or receivers for the topic, then this delete call returns TIBEMS\_ILLEGAL\_STATE.

**See Also**

[tibemsTemporaryTopic](#)

**tibemsSession\_GetAcknowledgeMode**

*Function*

**Purpose**

Get the acknowledge mode of a session.

**C Declaration**

```
tibems_status tibemsSession_GetAcknowledgeMode(
    tibemsSession session,
    tibemsAcknowledgeMode* acknowledgeMode );
```

## COBOL Call

```
CALL "tibemsSession_GetAcknowledgeMode"
    USING BY VALUE session,
          BY REFERENCE acknowledgeMode,
          RETURNING tibems-status
END-CALL.
```



session has usage pointer.

### Parameters

Parameter	Description
session	Get the property from this session.
acknowledgeMode	The function stores the property value in this location.

### Remarks

This mode governs message acknowledgement and redelivery for consumers associated with the session. For values, see [tibemsAcknowledgeMode](#).

This property is irrelevant when the session has transactional semantics.

## tibemsSession\_GetTransacted

*Function*

### Purpose

Get the transactional semantics property of a session.

### C Declaration

```
tibems_status tibemsSession_GetTransacted(
    tibemsSession session,
    tibems_bool* isTransacted );
```

## COBOL Call

```
CALL "tibemsSession_GetTransacted"
    USING BY VALUE session,
          BY REFERENCE isTransacted,
          RETURNING tibems-status
END-CALL.
```



session has usage pointer.

### Parameters

Parameter	Description
session	Get the property from this session.
isTransacted	The function stores the property value in this location.

## Remarks

When true, then the session has transaction semantics, and the session's acknowledge mode is irrelevant.

When false, it has non-transaction semantics.

## tibemsSession\_Recover

Function

### Purpose

Recover from undetermined state during message processing.

### C Declaration

```
tibems_status tibemsSession_Recover(
    tibemsSession session );
```

### COBOL Call

```
CALL "tibemsSession_Recover"
    USING BY VALUE session,
    RETURNING tibems-status
END-CALL.
```



session has usage pointer.

### Parameters

Parameter	Description
session	Recover this session.

### Remarks

Exceptions during message processing can sometimes leave a program in an ambiguous state. For example, some messages might be partially processed. This function lets a program return to an unambiguous state—the point within the message stream when the program last acknowledged the receipt of inbound messages. Programs can then review the messages delivered since that point (they are marked as *redelivered*), and resolve ambiguities about message processing. Programs can also use this function to resolve similar ambiguities after a [tibemsConnection](#) stops delivering messages, and then starts again.

### Operation

This function requests that the server:

1. Stop message delivery within the session.
2. Mark as *redelivered*, any messages that the server has attempted to deliver to the session, but for which it has not received acknowledgement (that is, messages for which processing state is ambiguous). According to the JMS specification, the server need not redeliver messages in the same order as it first delivered them.
3. Restart message delivery (including messages marked as *redelivered* in step 2).

## Transactions

Commit and rollback are more appropriate with transactions. When a session has transactional semantics, this call is illegal, and returns [TIBEMS\\_INVALID\\_SESSION](#).

### See Also

[tibemsMsg\\_Recover](#)

## tibemsSession\_Rollback

*Function*

### Purpose

Roll back messages in the current transaction.

### C Declaration

```
tibems_status tibemsSession_Rollback(
    tibemsSession session );
```

### COBOL Call

```
CALL "tibemsSession_Rollback"
    USING BY VALUE session,
    RETURNING tibems-status
END-CALL.
```



session has usage pointer.

### Parameters

Parameter	Description
session	Rollback the messages in this session.

### Remarks

When a session does not have transactional semantics, this function returns [TIBEMS\\_ILLEGAL\\_STATE](#).

Messages sent to a queue with `prefetch=none` and `maxRedelivery=number` properties are not received `number` times by an EMS application that receives in a loop and does an XA rollback after the XA prepare phase.

## tibemsSession\_Unsubscribe

*Function*

### Purpose

Unsubscribe a durable topic subscription.

### C Declaration

```
tibems_status tibemsSession_Unsubscribe(
    tibemsSession session
    const char* name );
```

## COBOL Call

```
CALL "tibemsSession_Unsubscribe"
    USING BY VALUE session,
          BY REFERENCE name,
          RETURNING tibems-status
END-CALL.
```



session has usage pointer.

### Parameters

Parameter	Description
session	Delete a subscription in this session.
name	Enable the server to locate the subscription.

### Remarks

This function deletes the subscription from the server.

You must unsubscribe *before* closing the session.



It is illegal to delete an active subscription—that is, while a `tibemsMsgConsumer` exists.

It is illegal to delete a subscription while one of its messages is either unacknowledged, or uncommitted (in the current transaction). Attempting to do so results in a status of `TIBEMS_EXCEPTION`.

### See Also

[tibemsMsgConsumer](#)

[tibemsTopic](#)

[tibemsSession\\_CreateDurableSubscriber](#)

## tibemsAcknowledgeMode

Type

### Purpose

Define acknowledgment mode constants.

Constant	Description
TIBEMS_AUTO_ACKNOWLEDGE	<p>In this mode, the session automatically acknowledges a message when message processing is finished—that is, when either of these calls returns successfully:</p> <ul style="list-style-type: none"> <li>• synchronous receive calls (such as <a href="#">tibemsMsgConsumer_Receive</a>)</li> <li>• asynchronous listener callback (namely, <a href="#">tibemsMsgCallback</a>)</li> </ul>

Constant	Description
TIBEMS_CLIENT_ACKNOWLEDGE	In this mode, the client program acknowledges receipt by calling <code>tibemsMsg_Acknowledge</code> . Each call acknowledges all messages received so far.
TIBEMS_DUPS_OK_ACKNOWLEDGE	As with <code>TIBEMS_AUTO_ACKNOWLEDGE</code> , the session automatically acknowledges messages. However, it may do so lazily.  <i>Lazy</i> means that the provider client library can delay transferring the acknowledgement to the server until a convenient time; meanwhile the server might redeliver the message. Lazy acknowledgement can reduce session overhead.
TIBEMS_EXPLICIT_CLIENT_ACKNOWLEDGE	As with <code>TIBEMS_CLIENT_ACKNOWLEDGE</code> , the client program acknowledges receipt by calling <code>tibemsMsg_Acknowledge</code> . However, each call acknowledges <i>only</i> the individual message. The client may acknowledge messages in any order.  This mode and behavior are proprietary extensions, specific to TIBCO EMS.
TIBEMS_EXPLICIT_CLIENT_DUPS_OK_ACKNOWLEDGE	In this mode, the client program lazily acknowledges <i>only</i> the individual message, by calling <code>tibemsMsg_Acknowledge</code> . The client may acknowledge messages in any order.  <i>Lazy</i> means that the provider client library can delay transferring the acknowledgement to the server until a convenient time; meanwhile the server might redeliver the message.  This mode and behavior are proprietary extensions, specific to TIBCO EMS.
TIBEMS_NO_ACKNOWLEDGE	In <code>TIBEMS_NO_ACKNOWLEDGE</code> mode, messages do not require acknowledgement (which reduces message overhead). The server never redelivers messages.  This mode and behavior are proprietary extensions, specific to TIBCO EMS.

## COBOL

```

01  TIBEMS-ACKNOWLEDGE-MODES .
    05  TIBEMS-AUTO-ACKNOWLEDGE      PIC S9(8) COMP VALUE  1 .
    05  TIBEMS-CLIENT-ACKNOWLEDGE   PIC S9(8) COMP VALUE  2 .
    05  TIBEMS-DUPS-OK-ACKNOWLEDGE  PIC S9(8) COMP VALUE  3 .
    05  TIBEMS-NO-ACKNOWLEDGE       PIC S9(8) COMP VALUE 22 .

```

05	TIBEMS-EXPLICIT-CL-ACK	PIC	S9(8)	COMP	VALUE	23.
05	TIBEMS-EXPLICIT-CL-DUPS-OK-ACK	PIC	S9(8)	COMP	VALUE	24.

# Queue Browser

Queue browsers let client programs examine the messages on a queue without removing them from the queue.

## tibemsQueueBrowser

*Type*

### Purpose

View the messages in a queue without consuming them.

### Remarks

A browser is a dynamic enumerator of the queue (not a static snapshot). The queue is at the server, and its contents change as message arrive and consumers remove them. Meanwhile, while the browser is at the client. The function `tibemsQueueBrowser_GetNext` gets the next message from the server.

The browser can enumerate messages in a queue, or a subset filtered by a message selector.

Sessions serve as factories for queue browsers; see `tibemsSession_CreateBrowser`.

Function	Description
<code>tibemsQueueBrowser_Close</code>	Close the browser; reclaim resources.
<code>tibemsQueueBrowser_GetNext</code>	Get the next message from the browser.
<code>tibemsQueueBrowser_GetMsgSelector</code>	Get the selector string for the browser.
<code>tibemsQueueBrowser_GetQueue</code>	Get the queue that the browser examines.

### See Also

`tibemsSession_CreateBrowser`

## tibemsQueueBrowser\_Close

*Function*

### Purpose

Close the browser; reclaim resources.

### C Declaration

```
tibems_status tibemsQueueBrowser_Close(
    tibemsQueueBrowser queueBrowser );
```

### COBOL Call

```
CALL "tibemsQueueBrowser_Close"
    USING BY VALUE queueBrowser,
    RETURNING tibems-status
END-CALL.
```



queueBrowser has usage pointer.

**Parameters**

Parameter	Description
queueBrowser	Close this queue browser.

**tibemsQueueBrowser\_GetMsgSelector**

Function

**Purpose**

Get the selector string for the browser.

**C Declaration**

```
tibems_status tibemsQueueBrowser_GetMsgSelector(
    tibemsQueueBrowser queueBrowser,
    const char** selector );
```

**COBOL Call**

```
CALL "tibemsQueueBrowser_GetMsgSelector"
    USING BY VALUE queueBrowser,
         BY REFERENCE selector,
         RETURNING tibems-status
END-CALL.
```



queueBrowser and selector have usage pointer.

**Parameters**

Parameter	Description
queueBrowser	Get the selector from the queue of this browser.
selector	The function stores the selector string in this location.

**Remarks**When non-null, the browser presents only messages that match this selector; see [Message Selectors](#).

When null, or the empty string, the browser views all messages in the queue.

**tibemsQueueBrowser\_GetNext**

Function

**Purpose**

Get the next message from the browser.

**C Declaration**

```
tibems_status tibemsQueueBrowser_GetNext(
    tibemsQueueBrowser queueBrowser,
    tibemsMsg* msg );
```

## COBOL Call

```
CALL "tibemsQueueBrowser_GetNext"
    USING BY VALUE queueBrowser,
          BY REFERENCE msg,
          RETURNING tibems-status
END-CALL.
```



queueBrowser and msg have usage pointer.

### Parameters

Parameter	Description
queueBrowser	Get the next message from the queue of this browser.
msg	Stores the next message.

### Remarks

A browser is a dynamic enumerator of the queue (not a static snapshot). The queue is at the server, and its contents change as message arrive and consumers remove them. Meanwhile, while the browser is at the client. This function gets the next message from the server.

If prior calls to this function have exhausted the messages in the queue, the function returns [TIBEMS\\_NOT\\_FOUND](#).

## tibemsQueueBrowser\_GetQueue

*Function*

### Purpose

Get the queue that the browser examines.

### C Declaration

```
tibems_status tibemsQueueBrowser_GetQueue(
    tibemsQueueBrowser queueBrowser,
    tibemsQueue* queue );
```

## COBOL Call

```
CALL "tibemsQueueBrowser_GetQueue"
    USING BY VALUE queueBrowser,
          BY REFERENCE queue,
          RETURNING tibems-status
END-CALL.
```



queueBrowser and queue have usage pointer.

### Parameters

Parameter	Description
queueBrowser	Get the browser queue.
queue	Stores the queue.

# Name Server Lookup

Lookup context objects find named objects (such as connection factories and destinations) in the name repository. (The EMS server, tibemsd, provides the name repository service).

## tibemsLookupContext

*Type*

### Purpose

Retrieve objects from the server's naming directory.

### Remarks

The context object establishes communication with an EMS server or an LDAP server, authenticates the user, and submits name queries.

Name queries can retrieve connection factories and destinations.

Function	Description
<a href="#">tibemsLookupContext_Create</a> tibemsLookupContext_CreateSSL	Create a new EMS lookup context object.
<a href="#">tibemsLookupContext_CreateExternal</a>	Create a new LDAP lookup context object.
<a href="#">tibemsLookupContext_Destroy</a>	Destroy a lookup context and reclaim resources.
<a href="#">tibemsLookupContext_Lookup</a> tibemsLookupContext_LookupDestination tibemsLookupContext_LookupConnectionFactory	Look up an object in the naming server.

## tibemsLookupContext\_Create

*Function*

### Purpose

Create a new EMS lookup context object.

### C Declaration

```

tibems_status tibemsLookupContext_Create(
    tibemsLookupContext* context,
    const char* brokerURL,
    const char* username,
    const char* password );

tibems_status tibemsLookupContext_CreateSSL(
    tibemsLookupContext* context,
    const char* brokerURL,
    const char* username,
    const char* password,
    tibemsSSLParams sslParams,
    const char* pk_password );

```

## COBOL Call

```
CALL "tibemsLookupContext_Create"
    USING BY REFERENCE context,
          BY REFERENCE brokerURL,
          BY REFERENCE username,
          BY REFERENCE password,
          RETURNING tibems-status
END-CALL.

CALL "tibemsLookupContext_CreateSSL"
    USING BY REFERENCE context,
          BY REFERENCE brokerURL,
          BY REFERENCE username,
          BY REFERENCE password,
          BY VALUE sslParams,
          BY REFERENCE pk-password,
          RETURNING tibems-status
END-CALL.
```



context and sslParams have usage pointer.

On IBM z/OS systems, the pk-password must always be a null value.

## Parameters

Parameter	Description
context	Store the new lookup context object in this location.
brokerURL	URL that the context object uses to connect to the EMS server at this URL. If configuring a fault-tolerant client, enter two or more URLs, as described in <a href="#">Configuring C and COBOL Clients for Fault-Tolerant Connections</a> .
username	User name that the context object uses to identify itself to the server.
password	Password that the context object uses to identify itself to the server.
sslParams	Data that the context object uses to create an SSL connection to the EMS server.
pk_password	Password that the context object uses to decrypt its SSL private key.

## Remarks

The first call produces a lookup context that communicates with server without encryption. The second call produces a lookup context that communicates using an SSL connection.

If the server permits anonymous lookup, you may supply null values for the username and password parameters.

## See Also

[tibemsLookupContext\\_CreateExternal](#)

## tibemsLookupContext\_CreateExternal

*Function*

### Purpose

Create a new LDAP lookup context object.

### C Declaration

```
tibems_status tibemsLookupContext_CreateExternal(
    tibemsLookupContext* context,
    tibemsLookupParams lookupParams );
```

### COBOL Call

```
CALL "tibemsLookupContext_CreateExternal"
    USING BY REFERENCE context,
         BY VALUE lookupParams,
         RETURNING tibems-status
END-CALL.
```



context and lookupParams have usage pointer.

### Parameters

Parameter	Description
context	Store the new lookup context object in this location.
lookupParams	The context object uses these parameters to connect to an LDAP server. After the function returns successfully, the calling program may delete this struct.

### See Also

[tibemsLookupContext\\_Create](#)

[tibemsLookupParams](#)

[tibemsLookupParams\\_Create](#)

[tibemsLookupParams\\_Destroy](#)

## tibemsLookupContext\_Destroy

*Function*

### Purpose

Destroy a lookup context and reclaim resources.

### C Declaration

```
tibems_status tibemsLookupContext_Destroy(
    tibemsLookupContext context );
```

### COBOL Call

```
CALL "tibemsLookupContext_Destroy"
    USING BY VALUE context,
```

RETURNING tibems-status  
END-CALL.



context has usage pointer.

### Parameters

Parameter	Description
context	Destroy this lookup context object.

## tibemsLookupContext\_Lookup

*Function*

### Purpose

Look up an object in the naming server.

### C Declaration

```
tibems_status tibemsLookupContext_Lookup(
    tibemsLookupContext context,
    const char* name,
    void** object);

tibems_status tibemsLookupContext_LookupDestination(
    tibemsLookupContext context,
    const char* name,
    tibemsDestination* destination);

tibems_status tibemsLookupContext_LookupConnectionFactory(
    tibemsLookupContext context,
    const char* name,
    tibemsConnectionFactory* factory);
```

### COBOL Call

```
CALL "tibemsLookupContext_Lookup"
    USING BY VALUE context,
         BY REFERENCE name,
         BY REFERENCE object,
         RETURNING tibems-status
END-CALL.

CALL "tibemsLookupContext_LookupDestination"
    USING BY VALUE context,
         BY REFERENCE name,
         BY REFERENCE destination,
         RETURNING tibems-status
END-CALL.

CALL "tibemsLookupContext_LookupConnectionFactory"
    USING BY VALUE context,
         BY REFERENCE name,
         BY REFERENCE factory,
         RETURNING tibems-status
END-CALL.
```



context, object, destination and factory have usage pointer.

## Parameters

Parameter	Description
context	Destroy this lookup context object.
name	Lookup this name.
object	Store the results of the lookup operation.
destination	
factory	

## Remarks

These calls look up names in either the name server portion of an EMS server, or in an LDAP server.

The first call looks up a generic object; the calling program must cast the result to the expect type. The other calls restrict lookup to either destinations or connection factories.

If the server does not find the name, this call returns `TIBEMS_NOT_FOUND`.

If the server finds both a topic and a queue with the same name, this call returns `TIBEMS_ILLEGAL_STATE`.

The calling program must destroy the resulting object when it is no longer needed.

## tibemsLookupParams

Type

### Purpose

Encapsulate parameters for LDAP lookup.

Function	Description
<code>tibemsLookupParams_Create</code>	Create a new LDAP lookup context object.
<code>tibemsLookupParams_Destroy</code>	Destroy a lookup parameters object.
<code>tibemsLookupParams_GetLdapServerUrl</code>	Get the LDAP server URL.
<code>tibemsLookupParams_SetLdapBaseDN</code>	Set the LDAP base distinguished name.
<code>tibemsLookupParams_SetLdapCAFile</code>	Set the LDAP CA certificate file.
<code>tibemsLookupParams_SetLdapCAPath</code>	Set the LDAP CA certificate directory pathname.
<code>tibemsLookupParams_SetLdapCertFile</code>	Set the LDAP client certificate file.
<code>tibemsLookupParams_SetLdapCiphers</code>	Set SSL ciphers.

Function	Description
<a href="#">tibemsLookupParams_SetLdapConnType</a>	Set the LDAP connection type.
<a href="#">tibemsLookupParams_SetLdapCredential</a>	Set the LDAP credential (password).
<a href="#">tibemsLookupParams_SetLdapKeyFile</a>	Set the LDAP client key file.
<a href="#">tibemsLookupParams_SetLdapPrincipal</a>	Set the LDAP principal (user name).
<a href="#">tibemsLookupParams_SetLdapRandFile</a>	Set a randomization data file.
<a href="#">tibemsLookupParams_SetLdapSearchScope</a>	Set the LDAP search scope.
<a href="#">tibemsLookupParams_SetLdapServerUrl</a>	Set the LDAP server URL.

## **tibemsLookupParams\_Create**

*Function*

### **Purpose**

Create a new lookup parameters object.

### **C Declaration**

```
tibemsLookupParams
tibemsLookupParams_Create(void);
```

### **IBM Systems**

This function is not supported on z/OS and IBM i systems.

### **See Also**

[tibemsLookupContext\\_CreateExternal](#)

## **tibemsLookupParams\_Destroy**

*Function*

### **Purpose**

Destroy a lookup parameters object.

### **C Declaration**

```
void tibemsLookupParams_Destroy(
tibemsLookupParams lparams );
```

### **IBM Systems**

This function is not supported on z/OS and IBM i systems.

**Parameters**

Parameter	Description
lparams	Stores the new lookup context object.

**tibemsLookupParams\_GetLdapServerUrl***Function***Purpose**

Get the LDAP server URL.

**C Declaration**

```
char* tibemsLookupParams_GetLdapServerUrl(
    tibemsLookupParams lparams);
```

**IBM Systems**

This function is not supported on z/OS and IBM i systems.

**Parameters**

Parameter	Description
lparams	Get the LDAP server URL from this external lookup context object.

**tibemsLookupParams\_SetLdapBaseDN***Function***Purpose**

Set the LDAP base distinguished name.

**C Declaration**

```
tibems_status tibemsLookupParams_SetLdapBaseDN(
    tibemsLookupParams lparams,
    const char* basedn );
```

**IBM Systems**

This function is not supported on z/OS and IBM i systems.

**Parameters**

Parameter	Description
lparams	Set the property in this parameter object.
basedn	Set the property to this distinguished name.

**Remarks**

This parameter is required for all LDAP lookup operations.

## tibemsLookupParams\_SetLdapCAFile

*Function*

### Purpose

Set the LDAP CA certificate file.

### C Declaration

```
tibems_status tibemsLookupParams_SetLdapCAFile(
    tibemsLookupParams lparams,
    const char* file );
```

### IBM Systems

This function is not supported on z/OS and IBM i systems.

### Parameters

Parameter	Description
lparams	Set the property in this parameter object.
path	Set the property to this filename.

### Remarks

The client program reads the CA certificate from this file.

## tibemsLookupParams\_SetLdapCAPath

*Function*

### Purpose

Set the LDAP CA certificate directory pathname.

### C Declaration

```
tibems_status tibemsLookupParams_SetLdapCAPath(
    tibemsLookupParams lparams,
    const char* path );
```

### IBM Systems

This function is not supported on z/OS and IBM i systems.

### Parameters

Parameter	Description
lparams	Set the property in this parameter object.
path	Set the property to this directory pathname.

**Remarks**

Set this parameter when several certificate files are grouped in one directory. This parameter specifies that directory. The client program reads certificate files from this directory.

**tibemsLookupParams\_SetLdapCertFile**

*Function*

**Purpose**

Set the LDAP client certificate file.

**C Declaration**

```
tibems_status tibemsLookupParams_SetLdapCertFile(
    tibemsLookupParams lparams,
    const char* file );
```

**IBM Systems**

This function is not supported on z/OS and IBM i systems.

**Parameters**

Parameter	Description
lparams	Set the property in this parameter object.
file	Set the property to this filename.

**Remarks**

The client program reads the client certificate from this file.

Set this parameter when the LDAP server requires clients to present certificates as identification.

**tibemsLookupParams\_SetLdapCiphers**

*Function*

**Purpose**

Set SSL ciphers.

**C Declaration**

```
tibems_status tibemsLookupParams_SetLdapCiphers(
    tibemsLookupParams lparams,
    const char* ciphers );
```

**IBM Systems**

This function is not supported on z/OS and IBM i systems.

**Parameters**

Parameter	Description
lparams	Set the property in this parameter object.

Parameter	Description
ciphers	<p>Set the property to specify these ciphers (as a colon-separated list).</p> <p>For example:</p> <pre>"EDH-RSA-DES-CBC3-SHA:EDH-DSS-DES-CBC3-SHA:DES-CBC3-SHA:DES-CBC3-MD5:DHE-DSS-RC4-SHA:IDEA-CBC-SHA:RC4-SHA:RC4-MD5:IDEA-CBC-MD5:RC2-CBC-MD5:RC4-MD5:RC4-64-MD5:EXP1024-DHE-DSS-RC4-SHA:EXP1024-RC4-SHA:EXP1024-DHE-DSS-DES-CBC-SHA:EXP1024-DES-CBC-SHA:EXP1024-RC2-CBC-MD5:EXP1024-RC4-MD5:EDH-RSA-DES-CBC-SHA:EDH-DSS-DES-CBC-SHA:DES-CBC-SHA:DES-CBC-MD5:EXP-EDH-RSA-DES-CBC-SHA:EXP-EDH-DSS-DES-CBC-SHA:EXP-DES-CBC-SHA:EXP-RC2-CBC-MD5:EXP-RC4-MD5:EXP-RC2-CBC-MD5:EXP-RC4-MD5"</pre>

### Remarks

Set this parameter when using secure LDAP connections.

The LDAP server configures a similar list of ciphers. SSL communication uses the strongest ciphers in the intersection of the server and client cipher lists.

## tibemsLookupParams\_SetLdapConnType

*Function*

### Purpose

Set the LDAP connection type.

### C Declaration

```
tibems_status tibemsLookupParams_SetLdapConnType(
    tibemsLookupParams lparams,
    const char* type );
```

### IBM Systems

This function is not supported on z/OS and IBM i systems.

### Parameters

Parameter	Description
lparams	Set the property in this parameter object.
type	<p>Set the property to this type.</p> <p>When this value is not set, it specifies a simple (non-secure) connection.</p> <p>Either "ldaps" or "startTLS" specify secure connections.</p>

## tibemsLookupParams\_SetLdapCredential

*Function*

### Purpose

Set the LDAP credential (password).

**C Declaration**

```
tibems_status tibemsLookupParams_SetLdapCredential(
    tibemsLookupParams lparams,
    const char* credential );
```

**IBM Systems**

This function is not supported on z/OS and IBM i systems.

**Parameters**

Parameter	Description
lparams	Set the property in this parameter object.
credential	Set the property to this credential (password).

**tibemsLookupParams\_SetLdapKeyFile**

*Function*

**Purpose**

Set the LDAP client key file.

**C Declaration**

```
tibems_status tibemsLookupParams_SetLdapKeyFile(
    tibemsLookupParams lparams,
    const char* file );
```

**IBM Systems**

This function is not supported on z/OS and IBM i systems.

**Parameters**

Parameter	Description
lparams	Set the property in this parameter object.
file	Set the property to this filename.

**Remarks**

The client program reads the private key for the client certificate from this (encrypted) file.

Set this parameter when the LDAP server requires clients to present certificates as identification.

**tibemsLookupParams\_SetLdapPrincipal**

*Function*

**Purpose**

Set the LDAP principal (user name).

## C Declaration

```
tibems_status tibemsLookupParams_SetLdapPrincipal(
    tibemsLookupParams lparams,
    const char* principal );
```

## IBM Systems

This function is not supported on z/OS and IBM i systems.

## Parameters

Parameter	Description
lparams	Set the property in this parameter object.
principal	Set the property to this principal (user name).

## Remarks

This parameter is required for all LDAP lookup operations.

## tibemsLookupParams\_SetLdapRandFile

*Function*

## Purpose

Set a randomization data file.

## C Declaration

```
tibems_status tibemsLookupParams_SetLdapRandFile(
    tibemsLookupParams lparams,
    const char* file );
```

## IBM Systems

This function is not supported on z/OS and IBM i systems.

## Parameters

Parameter	Description
lparams	Set the property in this parameter object.
file	Set the property to this filename.

## Remarks

The client program reads random data for security computations from this file.

Set this parameter when the operating system does not provide a random data service.

## tibemsLookupParams\_SetLdapSearchScope

*Function*

### Purpose

Set the LDAP search scope.

### C Declaration

```
tibems_status tibemsLookupParams_SetLdapSearchScope(
    tibemsLookupParams lparams,
    const char* scope );
```

### IBM Systems

This function is not supported on z/OS and IBM i systems.

### Parameters

Parameter	Description
lparams	Set the property in this parameter object.
scope	Set the property to this search scope.

### Remarks

This parameter is required for all LDAP lookup operations.

## tibemsLookupParams\_SetLdapServerUrl

*Function*

### Purpose

Set the LDAP server URL.

### C Declaration

```
tibems_status tibemsLookupParams_SetLdapServerUrl(
    tibemsLookupParams lparams,
    const char* url);
```

### IBM Systems

This function is not supported on z/OS and IBM i systems.

### Parameters

Parameter	Description
lparams	Set the property in this parameter object.
url	Set the property to this URL.

### Remarks

This parameter is required for all LDAP lookup operations.

# XA—External Transaction Manager

The EMS XA API lets you code your own transaction manager (TM) that can interact with the EMS server as a transactional resource.

## tibemsXAConnection

This implicit type is not defined; instead it is identical to [tibemsConnection](#). The functions we present on the following pages apply only to connection objects used with XA.

### tibemsXAConnection\_Close

*Function*

#### Purpose

Close the connection; reclaim resources.

#### C Declaration

```
tibems_status tibemsXAConnection_Close(
    tibemsConnection connection );
```

#### COBOL Call

```
CALL "tibemsXAConnection_Close"
    USING BY VALUE connection,
         RETURNING tibems-status
END-CALL.
```



connection has usage pointer.

#### Parameters

Parameter	Description
connection	Close this connection.

#### Remarks

Closing an XA connection reclaims all XA resources associated with the connection or its sessions.

Closing the connection is not sufficient to reclaim all of its resources; your program must explicitly close the sessions, producers, and consumers associated with the connection.

Closing a connection deletes all temporary destinations associated with the connection.

#### Blocking

If any message listener or receive call associated with the connection is processing a message when the program calls this function, all facilities of the connection and its sessions remain available to those listeners until they return. In the meantime, this function blocks until that processing completes—that is, until all message listeners and receive calls have returned.

#### Acknowledge

Closing a connection does *not* force acknowledgment in client-acknowledged sessions. When the program still has a message that it received from a connection that has since closed, [tibemsg\\_Acknowledge](#) returns the status code [TIBEMS\\_ILLEGAL\\_STATE](#).

#### Transactions

Closing a connection rolls back all open transactions in all sessions associated with the connection.

### See Also

[tibemsMsg\\_Acknowledge](#)

[tibemsMsgConsumer](#)

[tibemsMsgProducer](#)

[tibemsDestination](#)

[tibemsSession](#)

## tibemsXAConnection\_Create

*Function*

### Purpose

Create a new XA connection to an EMS server; use XA for transactions.

### C Declarations

```
tibems_status tibemsXAConnection_Create(
    tibemsConnection* connection,
    const char* brokerURL,
    const char* clientId,
    const char* username,
    const char* password );

tibems_status tibemsXAConnection_CreateSSL(
    tibemsConnection* connection,
    const char* brokerURL,
    const char* clientId,
    const char* username,
    const char* password,
    tibemsSSLParams sslParams,
    const char* pk_password );
```

### COBOL Call

```
CALL "tibemsXAConnection_Create"
    USING BY REFERENCE connection,
          BY REFERENCE brokerURL,
          BY REFERENCE clientId,
          BY REFERENCE username,
          BY REFERENCE password,
          RETURNING tibems-status
END-CALL.
```



On IBM z/OS systems, the pk-password must always be a null value.

### Parameters

Parameter	Description
connection	Store the new connection in this location.
brokerURL	URL location of the EMS server. If configuring a fault-tolerant client, enter two or more URLs, as described in <a href="#">Configuring C and COBOL Clients for Fault-Tolerant Connections</a> .
clientId	Identify the client program to the server with this unique ID.

Parameter	Description
username	Identify the client program to the server with this user name.
password	Authenticate the client program to the server with this password.
sslParams	Establish SSL communication using these parameters.
pk_password	Private key password for SSL.

Status Code	Description
<a href="#">TIBEMS_OK</a>	The call succeeded.
<a href="#">TIBEMS_SERVER_NOT_CONNECTED</a>	<ul style="list-style-type: none"> <li>No server is running at the specified URL.</li> <li>The call could not communicate with a server because of mismatched SSL and TCP protocols.</li> <li>Other error situations are possible.</li> </ul>
<a href="#">TIBEMS_SECURITY_EXCEPTION</a>	<ul style="list-style-type: none"> <li>The server rejected the connection because the username or password was invalid.</li> <li>SSL setup is incorrect.</li> </ul>
<a href="#">TIBEMS_INVALID_CLIENT_ID</a>	The client ID is not unique; that is, another client already uses the ID.

### See Also

[tibemsConnection\\_Create](#)

## tibemsXAConnection\_CreateXASession

*Function*

### Purpose

Create an XA session object.

### C Declaration

```
tibems_status tibemsXAConnection_CreateXASession(
    tibemsConnection connection,
    tibemsSession* session );
```

### COBOL Call

```
CALL "tibemsXAConnection_CreateXASession"
    USING BY VALUE connection,
          BY REFERENCE session,
          RETURNING tibems-status
END-CALL.
```

connection and session have usage pointer.



## Parameters

Parameter	Description
connection	Create a session on this connection.
session	Stores the new session.

## Remarks

The new session has transactional semantics with an external transaction manager, and uses the connection for all server communications.



XA sessions do not support routed queues.

## See Also

[tibemsSession](#)

## tibemsXAConnection\_Get

*Function*

### Purpose

Find the XA connection object for a server URL.

### C Declaration

```
tibems_status tibemsXAConnection_Get(
    tibemsConnection* connection,
    const char* brokerURL );
```

### COBOL Call

```
CALL "tibemsXAConnection_Get"
    USING BY REFERENCE connection,
         BY REFERENCE brokerURL,
         RETURNING tibems-status
END-CALL.
```



connection has usage pointer.

## Parameters

Parameter	Description
connection	Stores the connection object.
brokerURL	Find the connection the EMS server at this URL. If configuring a fault-tolerant client, enter two or more URLs, as described in <a href="#">Configuring C and COBOL Clients for Fault-Tolerant Connections</a> .

## Remarks

If the TM has implicitly created a connection by calling `xa_open`, then the TM can get that connection object with this call.

## tibemsXAConnection\_GetXASession

*Function*

### Purpose

Get the XA session object from an XA connection.

### C Declaration

```
tibems_status tibemsXAConnection_GetXASession(
    tibemsConnection* connection,
    tibemsSession* xaSession );
```

### COBOL Call

```
CALL "tibemsXAConnection_GetXASession"
    USING BY VALUE connection,
         BY REFERENCE xaSession,
         RETURNING tibems-status
END-CALL.
```



connection and xaSession have usage pointer.

### Parameters

Parameter	Description
connection	Get the XA session object from this XA connection.
xaSession	Stores the XA session object in this location.

### Remarks

If the TM has implicitly created a session by calling xa\_open, then the TM can get that session object with this call.

## XID

Type

### Purpose

Represent a transaction ID.

## tibemsXAResource

Type

### Purpose

Coordinate XA transactions.

### Remarks

Each tibemsXAResource instance can coordinate a series of transactions, but only one transaction at a time. A program that keeps  $n$  transactions open simultaneously must create  $n$  instances of this type.

The transaction manager assigns each resource instance a unique RMID, which it uses to bind together a thread, a resource and a transaction.

Function	Description
<code>tibemsXAResource_Commit</code>	Commit a transaction.
<code>tibemsXAResource_End</code>	Disassociate a transaction from the resource.
<code>tibemsXAResource_Forget</code>	Unimplemented.
<code>tibemsXAResource_GetRMID</code>	Get the RMID of the resource.
<code>tibemsXAResource_GetTransactionTimeout</code>	Get the timeout limit.
<code>tibemsXAResource_isSameRM</code>	Determine whether two resource objects originate from the same connection to an EMS server.
<code>tibemsXAResource_Prepare</code>	Prepare a transaction for commit.
<code>tibemsXAResource_Recover</code>	Get a list of prepared transactions.
<code>tibemsXAResource_Rollback</code>	Roll back a transaction.
<code>tibemsXAResource_SetRMID</code>	Assign an RMID to a resource.
<code>tibemsXAResource_SetTransactionTimeout</code>	Set the timeout limit.
<code>tibemsXAResource_Start</code>	Associate a transaction with a resource.

## **tibemsXAResource\_Commit**

Function

### **Purpose**

Commit a transaction.

### **C Declaration**

```
tibems_status tibemsXAResource_Commit(
    tibemsXAResource xaResource,
    XID* xid,
    tibems_bool onePhase );
```

### **COBOL Call**

```
CALL "tibemsXAResource_Commit"
    USING BY VALUE xaResource,
          BY REFERENCE xid,
          BY VALUE onePhase,
          RETURNING tibems-status
END-CALL.
```



`xaResource` and `xid` have usage pointer.

## Parameters

Parameter	Description
xaResource	Commit the transaction in this resource.
xid	Commit this transaction.
onePhase	TIBEMS_TRUE requests a <i>one-phase commit</i> . TIBEMS_FALSE requests a <i>two-phase commit</i> .

## tibemsXAResource\_End

Function

### Purpose

Disassociate a transaction from the resource.

### C Declaration

```
tibems_status tibemsXAResource_End(
    tibemsXAResource xaResource,
    XID* xid,
    int flags );
```

### COBOL Call

```
CALL "tibemsXAResource_End"
    USING BY VALUE xaResource,
          BY REFERENCE xid,
          BY VALUE flags,
          RETURNING tibems-status
END-CALL.
```



xaResource and xid have usage pointer.

## Parameters

Parameter	Description
xaResource	Disassociate the transaction from this resource.
xid	Disassociate this transaction from the resource.
flags	TMSUCCESS—Completely end the transaction. TMSUSPEND—Temporarily disassociate the transaction from this resource.

## tibemsXAResource\_Forget

Function

### Purpose

Unimplemented.

## C Declaration

```
tibems_status tibemsXAResource_Forget(
    tibemsXAResource xaResource,
    XID* xid );
```

## COBOL Call

```
CALL "tibemsXAResource_Forget"
    USING BY VALUE xaResource,
          BY REFERENCE xid,
          RETURNING tibems-status
END-CALL.
```



xaResource and xid have usage pointer.

## Parameters

Parameter	Description
xaResource	Forget the transaction in this resource.
xid	Forget this transaction.

## Remarks

In the XA interface, the transaction manager can instruct a resource to forget a transaction.

However, this call is not implemented in EMS; it is present for completeness only.

## tibemsXAResource\_GetRMID

Function

### Purpose

Get the RMID of the resource.

## C Declaration

```
tibems_status tibemsXAResource_GetRMID(
    tibemsXAResource xaResource,
    tibems_int* rmid );
```

## COBOL Call

```
CALL "tibemsXAResource_GetRMID"
    USING BY VALUE xaResource,
          BY REFERENCE rmid,
          RETURNING tibems-status
END-CALL.
```



xaResource has usage pointer.

## Parameters

Parameter	Description
xaResource	Get the RMID of this resource.

Parameter	Description
rmid	Store the RMID in this location.

### Remarks

The transaction manager assigns a unique RMID to each resource instance, which it uses to bind together a thread, a resource and a transaction.

## tibemsXAResource\_GetTransactionTimeout

Function

### Purpose

Get the timeout limit.

### C Declaration

```
tibems_status tibemsXAResource_GetTransactionTimeout(
    tibemsXAResource xaResource,
    tibems_int* seconds );
```

### COBOL Call

```
CALL "tibemsXAResource_GetTransactionTimeout"
    USING BY VALUE xaResource,
          BY REFERENCE seconds,
          RETURNING tibems-status
END-CALL.
```



xaResource has usage pointer.

### Parameters

Parameter	Description
xaResource	Get the transaction timeout of this resource.
seconds	Store the transaction timeout (in seconds).

### Remarks

If an open, unprepared transaction does not log any activity on the server for this length of time, then the server automatically rolls back the transaction.

### See Also

[tibemsXAResource\\_SetTransactionTimeout](#)

## tibemsXAResource\_isSameRM

Function

### Purpose

Determine whether two resource objects originate from the same connection to an EMS server.

## C Declaration

```
tibems_status tibemsXAResource_isSameRM(
    tibemsXAResource xaResource,
    tibemsXAResource xaResource2,
    tibems_bool* result );
```

## COBOL Call

```
CALL "tibemsXAResource_isSameRM"
    USING BY VALUE xaResource,
          BY VALUE xaResource2,
          BY REFERENCE result,
          RETURNING tibems-status
END-CALL.
```



xaResource and xaResource2 have usage pointer.

## Parameters

Parameter	Description
xaResource	First XA resource.
xaResource2	Second XA resource.
result	Store the result.

## tibemsXAResource\_Prepare

Function

### Purpose

Prepare a transaction for commit.

## C Declaration

```
tibems_status tibemsXAResource_Prepare(
    tibemsXAResource xaResource,
    XID* xid );
```

## COBOL Call

```
CALL "tibemsXAResource_Prepare"
    USING BY VALUE xaResource,
          BY REFERENCE xid,
          RETURNING tibems-status
END-CALL.
```



xaResource and xid have usage pointer.

## Parameters

Parameter	Description
xaResource	Prepare the transaction in this resource.
xid	Prepare this transaction.

## tibemsXAResource\_Recover

Function

### Purpose

Get a list of prepared transactions.

### C Declaration

```
tibems_status tibemsXAResource_Recover(
    tibemsXAResource xaResource,
    XID* xids,
    tibems_int desiredCount,
    tibems_int* returnedCount,
    tibems_int flag );
```

### COBOL Call

```
CALL "tibemsXAResource_Recover"
    USING BY VALUE xaResource,
          BY REFERENCE xids,
          BY VALUE desiredCount,
          BY REFERENCE returnedCount,
          BY VALUE flag,
          RETURNING tibems-status
END-CALL.
```



xaResource and xids have usage pointer.

### Parameters

Parameter	Description
xaResource	List the prepared transactions of this resource.
xids	Store the list of transaction IDs in the array.
desiredCount	Size of the array (number of XIDs).
returnedCount	Store the actual number of transaction.
flag	<p>TMSTARTSCAN—Start a new list of XIDs; the EMS server generates a complete list, and sends the first batch.</p> <p>TMNOFLAGS—Continue the list of XIDs; the EMS server sends the next batch.</p> <p>TMENDRSCAN—The EMS server discards its list of prepared transactions, and reclaims storage.</p>

### Remarks

When this call returns, if returnedCount = desiredCount, then more prepared transactions might exist. To get the next batch of XIDs, call this function again with TMNOFLAGS flag until returnedCount < desiredCount.

## tibemsXAResource\_Rollback

Function

### Purpose

Roll back a transaction.

### C Declaration

```
tibems_status tibemsXAResource_Rollback(
    tibemsXAResource xaResource,
    XID* xid );
```

### COBOL Call

```
CALL "tibemsXAResource_Rollback"
    USING BY VALUE xaResource,
          BY REFERENCE xid,
          RETURNING tibems-status
END-CALL.
```



xaResource and xid have usage pointer.

### Parameters

Parameter	Description
xaResource	Roll back in this resource.
xid	Roll back this transaction.

### Remarks

Messages sent to a queue with prefetch=none and maxRedelivery=*number* properties are not received *number* times by an EMS application that receives in a loop and does an XA rollback after the XA prepare phase.

## tibemsXAResource\_SetRMID

Function

### Purpose

Assign an RMID to a resource.

### C Declaration

```
tibems_status tibemsXAResource_SetRMID(
    tibemsXAResource xaResource,
    tibems_int rmid );
```

### COBOL Call

```
CALL "tibemsXAResource_SetRMID"
    USING BY VALUE xaResource,
          BY VALUE rmid,
          RETURNING tibems-status
END-CALL.
```



xaResource has usage pointer.

**Parameters**

Parameter	Description
xaResource	Set the RMID of this resource.
rmid	Use this RMID.

**tibemsXAResource\_SetTransactionTimeout**

Function

**Purpose**

Set the timeout limit.

**C Declaration**

```
tibems_status tibemsXAResource_SetTransactionTimeout(
    tibemsXAResource xaResource,
    tibems_int seconds );
```

**COBOL Call**

```
CALL "tibemsXAResource_SetTransactionTimeout"
    USING BY VALUE xaResource,
          BY VALUE seconds,
          RETURNING tibems-status
END-CALL.
```



xaResource has usage pointer.

**Parameters**

Parameter	Description
xaResource	Set the transaction timeout of this resource.
seconds	Use this transaction timeout (in seconds).

**Remarks**

If an unprepared transaction does not log any activity on the server for this length of time, then the server automatically rolls back the transaction.

**See Also**

[tibemsXAResource\\_GetTransactionTimeout](#)

**tibemsXAResource\_Start**

Function

**Purpose**

Associate a transaction with a resource.

## C Declaration

```
tibems_status tibemsXAResource_Start(
    tibemsXAResource xaResource,
    XID* xid,
    tibems_int flags );
```

## COBOL Call

```
CALL "tibemsXAResource_Start"
    USING BY VALUE xaResource,
          BY REFERENCE xid,
          BY VALUE flags,
          RETURNING tibems-status
END-CALL.
```



xaResource and xid have usage pointer.

## Parameters

Parameter	Description
xaResource	Associate a transaction with this resource.
xid	Associate this transaction with a resource.
flags	<p>TMNOFLAGS—This is a new transaction, which was not previously associated with the resource.</p> <p>TMRESUME—This transaction was previously associated with the resource.</p>

## tibemsXASession

Type

### Purpose

A session with an XA resource.

Function	Description
<a href="#">tibemsXASession_Close</a>	Close an XA session.
<a href="#">tibemsXASession_GetSession</a>	Get the EMS session from an XA session.
<a href="#">tibemsXASession_GetXAResource</a>	Get the XA resource of an XA session.



XA sessions do not support routed queues.

### See Also

[tibemsSession](#)

## tibemsXASession\_Close

Function

### Purpose

Close an XA session.

### C Declaration

```
tibems_status tibemsXASession_Close(
    tibemsSession session );
```

### COBOL Call

```
CALL "tibemsXASession_Close"
    USING BY VALUE session,
         RETURNING tibems-status
END-CALL.
```



session has usage pointer.

### Parameters

Parameter	Description
session	Close this session.

## tibemsXASession\_GetSession

Function

### Purpose

Get the EMS session from an XA session.

### C Declaration

```
tibems_status tibemsXASession_GetSession(
    tibemsSession xaSession,
    tibemsSession* session );
```

### COBOL Call

```
CALL "tibemsXASession_GetSession"
    USING BY VALUE xaSession,
         BY REFERENCE session,
         RETURNING tibems-status
END-CALL.
```



xaSession and session have usage pointer.

### Parameters

Parameter	Description
xaSession	Get the EMS session from this XA session.
session	Store the EMS session.

## tibemsXASession\_GetXAResource

Function

### Purpose

Get the XA resource of an XA session.

### C Declaration

```
tibems_status tibemsXASession_GetXAResource(
    tibemsSession session,
    tibemsXAResource* xaResource );
```

### COBOL Call

```
CALL "tibemsXASession_GetXAResource"
    USING BY VALUE session,
         BY REFERENCE xaResource,
         RETURNING tibems-status
END-CALL.
```



session and xaResource have usage pointer.

### Parameters

Parameter	Description
session	Get the XA resource from this session.
xaResource	Store the resource.

# Types

This section presents types in general use.

## Uniform Types

### Purpose

Standard datatypes.

### C Declarations

```
typedef char tibems_byte;
typedef short tibems_short;
typedef unsigned short tibems_wchar;
typedef int tibems_int;
typedef TIBX_I64 tibems_long;

typedef float tibems_float;
typedef double tibems_double;

typedef unsigned int tibems_uint;

typedef enum {
    TIBEMS_FALSE = 0,
    TIBEMS_TRUE = 1
} tibems_bool;
```

### EMS Types

Type	
tibems_byte	
tibems_short	
tibems_wchar	wide character; 2 bytes
tibems_int	
tibems_long	
tibems_float	
tibems_double	
tibems_uint	
tibems_bool	TIBEMS_FALSE, TIBEMS_TRUE

## Utilities

This section describes utility functions.

### Utility Functions

The functions described in this section are not related to a specific data type.

Function	Description
<code>tibems_Close</code>	Deallocate memory used by the EMS internal global data structures.
<code>tibems_GetAllowCloseInCallback</code>	Get whether the client application is allowed to call close or stop functions inside message listener callbacks.
<code>tibems_GetConnectAttemptCount</code>	Return the connection attempts limit.
<code>tibems_GetConnectAttemptDelay</code>	Return the connection delay.
<code>tibems_GetConnectAttemptTimeout</code>	Return the connection timeout.
<code>tibems_GetExceptionOnFTEvents</code>	Detect whether exception listener is called on any of the following fault-tolerant events: disconnected, each reconnect attempt, reconnected.
<code>tibems_GetExceptionOnFTSwitch</code>	Detect whether exception listener is called on a fault-tolerant switchover.
<code>tibems_GetMulticastDaemon</code>	Get the port of the multicast daemon that this application connects to.
<code>tibems_GetMulticastEnabled</code>	Get whether message consumers subscribed to multicast-enabled topics in EMS client applications can receive messages over multicast.
<code>tibems_GetReconnectAttemptCount</code>	Return the reconnection attempts limit.
<code>tibems_GetReconnectAttemptDelay</code>	Return the reconnection delay.
<code>tibems_GetReconnectAttemptTimeout</code>	Return the reconnection timeout.
<code>tibems_GetSocketReceiveBufferSize</code>	Return the size of socket receive buffers.
<code>tibems_GetSocketSendBufferSize</code>	Return the size of socket send buffers.
<code>tibems_IsConsumerMulticast</code>	Check if a consumer can receive messages over multicast.

Function	Description
<code>tibems_Open</code>	Allocate global memory for the EMS library.
<code>tibems_SetAllowCloseInCallback</code>	Set whether client applications can close sessions or connections, or stop connections, inside message listener callbacks.
<code>tibems_SetConnectAttemptCount</code>	Modify the connection attempt limit.
<code>tibems_SetConnectAttemptDelay</code>	Modify the connection delay.
<code>tibems_SetConnectAttemptTimeout</code>	Modify the connection timeout.
<code>tibems_SetExceptionOnFTEvents</code>	Set whether exception listener is called each step of the fault-tolerant switchover process.
<code>tibems_setExceptionOnFTSwitch</code>	Set whether exception listener is called on fault-tolerant switchover.
<code>tibems_SetMulticastDaemon</code>	Set the port on which the EMS client will connect to the multicast daemon.
<code>tibems_SetMulticastEnabled</code>	Set whether message consumers subscribed to multicast-enabled topics in the EMS client application will receive messages over multicast.
<code>tibems_SetMulticastExceptionListener</code>	Set the multicast exception listener for an application.
<code>tibems_SetReconnectAttemptCount</code>	Modify the reconnection attempts limit.
<code>tibems_SetReconnectAttemptDelay</code>	Modify the reconnection delay.
<code>tibems_SetReconnectAttemptTimeout</code>	Modify the reconnection timeout.
<code>tibems_SetSocketReceiveBufferSize</code>	Set the size of socket receive buffers.
<code>tibems_SetSocketSendBufferSize</code>	Set the size of socket send buffers.
<code>tibems_SetTraceFile</code>	Start or stop directing client trace information to a file.
<code>tibems_Sleep</code>	Sleep thread.
<code>tibems_Version</code>	Return the EMS library version number.

## **tibems\_Close**

*Function*

### **Purpose**

Deallocate the memory used by the EMS internal global data structures.

## C Declaration

```
void tibems_close(void);
```

## COBOL Call

```
CALL "tibems_close"  
  RETURNING tibems-Return-Value  
END-CALL.
```

## Remarks

Before unloading the EMS library, you must call this function to clean up. Windows platforms allow selective unloading of DLLs.

Applications that call [tibems\\_Open](#) and `tibems_Close` one or more times will allocate and deallocate correctly as long as the close calls match the open calls.

## See Also

[tibems\\_Open](#)

## `tibems_GetAllowCloseInCallback`

*Function*

## Purpose

Get whether the client application is allowed to call close or stop functions inside message listener callbacks.

## C Declaration

```
tibems_status tibems_GetAllowCloseInCallback(  
  tibems_bool *isAllowed);
```

## COBOL Call

```
CALL "tibems_GetAllowCloseInCallback"  
  USING BY REFERENCE isAllowed,  
  RETURNING tibems-status  
END-CALL.
```

## Parameters

Parameter	Description
<code>isAllowed</code>	Return <code>true</code> when the client application does not return with a <code>TIBEMS_ILLEGAL_STATE</code> error status before closing a session or connection or stopping a connection inside a message listener callback. Return <code>false</code> if the application returns with this error in such a situation.

## See Also

[tibems\\_SetAllowCloseInCallback](#)

## tibems\_GetConnectAttemptCount

*Function*

### Purpose

Return the connection attempts limit.

### C Declaration

```
tibems_int tibems_GetConnectAttemptCount(void);
```

### COBOL Call

```
CALL "tibems_GetConnectAttemptCount"  
  RETURNING tibems-Return-Value  
  END-CALL.
```

### Remarks

This setting governs all client [tibemsConnection](#) objects, limiting the number of times that connection objects attempt to establish a connection to the server. The default value is 2. The minimum value is 1.

### See Also

[tibems\\_SetConnectAttemptCount](#)

*Set Connection Attempts, Timeout and Delay Parameters* in the *TIBCO Enterprise Message Service User's Guide*

## tibems\_GetConnectAttemptDelay

*Function*

### Purpose

Return the connection delay.

### C Declaration

```
tibems_int tibems_GetConnectAttemptDelay(void);
```

### COBOL Call

```
CALL "tibems_GetConnectAttemptDelay"  
  RETURNING tibems-Return-Value  
  END-CALL.
```

### Remarks

This setting governs all client [tibemsConnection](#) objects. It determines the delay time between successive attempt to establish a connection to the server. Its value is the time (in milliseconds) between connection attempts. The default value is 500. The minimum value is 250.

### See Also

[tibems\\_SetConnectAttemptDelay](#)

*Set Connection Attempts, Timeout and Delay Parameters* in the *TIBCO Enterprise Message Service User's Guide*

## tibems\_GetConnectAttemptTimeout

*Function*

### Purpose

Return the connection timeout.

### C Declaration

```
tibems_int tibems_GetConnectAttemptTimeout(void);
```

### COBOL Call

```
CALL "tibems_GetConnectAttemptTimeout"  
  RETURNING tibems-Return-Value  
END-CALL.
```

### Remarks

This setting governs all client [tibemsConnection](#) objects. It determines the maximum time (in milliseconds) the client allows to establish a connection to the server. The default value is 0, indicating no timeout.

### See Also

[tibems\\_SetConnectAttemptTimeout](#)

*Set Connection Attempts, Timeout and Delay Parameters* in the *TIBCO Enterprise Message Service User's Guide*.

## tibems\_GetExceptionOnFTEvents

*Function*

### Purpose

Detect whether exception listener is called on any of the following fault-tolerant events: disconnected, each reconnect attempt, reconnected.

### C Declaration

```
tibems_status tibems_GetExceptionOnFTEvents(  
  tibems_bool* isSet);
```

### COBOL Call

```
CALL "tibems_GetExceptionOnFTEvents"  
  USING BY REFERENCE isSet,  
  RETURNING tibems-status  
END-CALL.
```

### Parameters

Parameter	Description
isSet	Return true if the exception listener is to be called on any of the following fault-tolerant events: disconnected, each reconnect attempt, reconnected. Return false if it is not to be called. Default is false.

**See Also**

[tibems\\_SetExceptionOnFTEvents](#)

Set an Exception Listener in the TIBCO Enterprise Message Service User's Guide.

**tibems\_GetExceptionOnFTSwitch**

Function

**Purpose**

Detect whether exception listener is called on a fault-tolerant switchover.

**C Declaration**

```
tibems_status tibems_GetExceptionOnFTSwitch(
    tibems_bool* isSet);
```

**COBOL Call**

```
CALL "tibems_GetExceptionOnFTSwitch"
    USING BY REFERENCE isSet,
    RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
isSet	Return true if the exception listener is to be called on a fault-tolerant switchover and false if it is not to be called. Default is false.

**See Also**

[tibems\\_setExceptionOnFTSwitch](#)

Set an Exception Listener in the TIBCO Enterprise Message Service User's Guide.

**tibems\_GetMulticastDaemon**

Function



Along with the multicast feature, this function was deprecated in software release 8.3.0 and will no longer be supported in future releases.

**Purpose**

Get the port of the multicast daemon that this application connects to.

**C Declaration**

```
const char* tibems_GetMulticastDaemon(void);
```

**Remarks**

Get the port of the multicast daemon that this application connects to.

This call is not supported in COBOL, and is not supported on z/OS and IBM i systems.

**See Also**

[tibems\\_SetMulticastDaemon](#)

**tibems\_GetMulticastEnabled**

*Function*



Along with the multicast feature, this function was deprecated in software release 8.3.0 and will no longer be supported in future releases.

**Purpose**

Get whether message consumers subscribed to multicast-enabled topics in EMS client applications can receive messages over multicast.

**C Declaration**

```
tibems_bool tibems_GetMulticastEnabled(void);
```

**Remarks**

Get whether message consumers subscribed to multicast-enabled topics in EMS client applications can receive messages over multicast.

This call is not supported in COBOL, and is not supported on z/OS and IBM i systems.

**See Also**

[tibems\\_SetMulticastEnabled](#)

[tibemsConnectionFactory\\_SetMulticastEnabled](#)

**tibems\_GetReconnectAttemptCount**

*Function*

**Purpose**

Return the reconnection attempts limit.

**C Declaration**

```
tibems_int tibems_GetReconnectAttemptCount(void);
```

**COBOL Call**

```
CALL "tibems_GetReconnectAttemptCount"
  RETURNING tibems-Return-Value
END-CALL.
```

**Remarks**

This setting governs all client [tibemsConnection](#) objects limiting the number of times that they attempt to reconnect to the server after a network disconnect. The default value is 4. The minimum value is 1.

**See Also**

[tibems\\_GetConnectAttemptCount](#)

[tibems\\_SetReconnectAttemptCount](#)

See *Set Reconnection Failure Parameters* in the *TIBCO Enterprise Message Service User's Guide*.

## tibems\_GetReconnectAttemptDelay

*Function*

### Purpose

Return the reconnection delay.

### C Declaration

```
tibems_int tibems_GetReconnectAttemptDelay(void);
```

### COBOL Call

```
CALL "tibems_GetReconnectAttemptDelay"  
  RETURNING tibems-Return-Value  
  END-CALL.
```

### Remarks

This setting governs all client [tibemsConnection](#) objects. It determines the delay time between successive attempt to establish a connection to the server. Its value is the time (in milliseconds) between connection attempts. The default value is 500. The minimum value is 250.

### See Also

[tibems\\_GetConnectAttemptDelay](#)

[tibems\\_SetReconnectAttemptDelay](#)

*Set Reconnection Failure Parameters* in the *TIBCO Enterprise Message Service User's Guide*.

## tibems\_GetReconnectAttemptTimeout

*Function*

### Purpose

Return the reconnection timeout.

### C Declaration

```
tibems_int tibems_GetReconnectAttemptTimeout(void);
```

### COBOL Call

```
CALL "tibems_GetReconnectAttemptTimeout"  
  RETURNING tibems-Return-Value  
  END-CALL.
```

### Remarks

This setting governs all client [tibemsConnection](#) objects. It determines the maximum time (in milliseconds) to client will allow to reestablish a connection to the server. The default value is 0, indicating no timeout. The minimum value is 250.

### See Also

[tibems\\_SetReconnectAttemptTimeout](#)

*Set Reconnection Failure Parameters* in the *TIBCO Enterprise Message Service User's Guide*.

## tibems\_GetSocketReceiveBufferSize

*Function*

### Purpose

Return the size of socket receive buffers.

### C Declaration

```
tibems_int tibems_GetSocketReceiveBufferSize(void);
```

### COBOL Call

```
CALL "tibems_GetSocketReceiveBufferSize"  
  RETURNING tibems-Return-Value  
  END-CALL.
```

### Remarks

When set, this value overrides the operating system's default for the size of receive buffers associated with sockets that the client uses for connections to the server. (Some operating systems do not allow you to override the default size.)

### See Also

[tibems\\_SetSocketReceiveBufferSize](#)

## tibems\_GetSocketSendBufferSize

*Function*

### Purpose

Return the size of socket send buffers.

### C Declaration

```
tibems_int tibems_GetSocketSendBufferSize(void);
```

### COBOL Call

```
CALL "tibems_GetSocketSendBufferSize"  
  RETURNING tibems-Return-Value  
  END-CALL.
```

### Remarks

When set, this value overrides the operating system's default for the size of send buffers associated with sockets that the client uses for connections to the server. (Some operating systems do not allow you to override the default size.)

### See Also

[tibems\\_SetSocketSendBufferSize](#)

## tibems\_IsConsumerMulticast

*Function*



Along with the multicast feature, this function was deprecated in software release 8.3.0 and will no longer be supported in future releases.

**Purpose**

Check if a consumer can receive messages over multicast.

**C Declaration**

```
tibems_bool tibems_IsConsumerMulticast(
    tibemsMsgConsumer consumer,
    tibems_bool *isMulticast);
```

**Remarks**

It is possible for a consumer to be a multicast consumer yet only receive messages over TCP. This may happen when the consumer is created on a wildcard destination that includes both unicast and multicast topics.

This call is not supported in COBOL, and is not supported on z/OS and IBM i systems.

**See Also**

[tibems\\_SetSocketSendBufferSize](#)

**tibems\_Open**

*Function*

**Purpose**

Allocate the global memory for the EMS library.

**C Declaration**

```
void tibems_Open(void);
```

**COBOL Call**

```
CALL "tibems_Open"
    RETURNING tibems-Return-Value
END-CALL.
```

**Remarks**

Applications that do not call `tibems_Open` will work as expected because the EMS internal global data structures are allocated when any public EMS function is called. However, the EMS data structures will not be deallocated until the application calls the `tibems_Close` function or exits.

Existing Windows applications that only call the `tibems_Close` function will work as expected.

Applications that call `tibems_Open` and `tibems_Close` one or more times will allocate and deallocate correctly as long as the close calls match the open calls.

**See Also**

[tibems\\_Close](#)

**tibems\_SetAllowCloseInCallback**

*Function*

**Purpose**

Set whether client applications can close sessions or connections, or stop connections, inside message listener callbacks.

## C Declaration

```
tibems_status tibems_SetAllowCloseInCallback(
    tibems_bool allow);
```

## COBOL Call

```
CALL "tibems_SetAllowCloseInCallback",
    USING BY VALUE isAllowed,
    RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
allow	<p>When true, the application will proceed without waiting for the message callbacks to return before closing a session or a connection or before stopping a connection. When false, the application will get a TIBEMS_ILLEGAL_STATE error status.</p> <p>By default, the application gets a TIBEMS_ILLEGAL_STATE status.</p>

## Remarks

This function can be used to change the default EMS client behavior and set whether the client application can call the `tibemsSession_Close`, `tibemsConnection_Close` or `tibemsConnection_Stop` functions from the `tibemsMsgCallback`.



According to the JMS 2.0 specification, calling any of these functions is not allowed and must return an error. If calling them inside the callback is allowed, then these functions when called inside the callback do not return an error and will proceed without waiting for the same callback to return.

The default behavior is to return with a TIBEMS\_ILLEGAL\_STATE status and parallel the behavior specified in the JMS 2.0 specification, which the EMS client API adheres to. However, this function allows applications to override the default behavior and close or stop EMS objects inside message callbacks.

## See Also

[tibemsMsgCallback](#)  
[tibemsMsgConsumer\\_Close](#)  
[tibemsConnection\\_Close](#)  
[tibemsSession\\_Close](#)  
[tibems\\_GetAllowCloseInCallback](#)

## tibems\_SetConnectAttemptCount

*Function*

### Purpose

Modify the connection attempt limit.

### C Declaration

```
tibems_status tibems_SetConnectAttemptCount(
    tibems_int count);
```

## COBOL Call

```
CALL "tibems_SetConnectAttemptCount"
  USING BY VALUE count,
  RETURNING tibems-status
END-CALL.
```

### Remarks

This setting governs all client [tibemsConnection](#) objects, limiting the number of times that connection objects attempt to establish a connection to the server.

### Parameters

Parameter	Description
count	Set the connect attempt limit to this value. The default value is 2. The minimum value is 1.

### See Also

[tibems\\_GetConnectAttemptCount](#)

*Set Connection Attempts, Timeout and Delay Parameters in the TIBCO Enterprise Message Service User's Guide.*

## tibems\_SetConnectAttemptDelay

*Function*

### Purpose

Modify the connection delay.

### C Declaration

```
tibems_status tibems_SetConnectAttemptDelay(
  tibems_int delay);
```

## COBOL Call

```
CALL "tibems_SetConnectAttemptDelay"
  USING BY VALUE delay,
  RETURNING tibems-status
END-CALL.
```

### Remarks

This setting governs all client [tibemsConnection](#) objects. It determines the delay time between successive attempt to establish a connection to the server. Its value is the time (in milliseconds) between connection attempts.

### Parameters

Parameter	Description
delay	Set the connect attempt delay to this time (in milliseconds). The default value is 500. The minimum value is 250.

**See Also**

[tibems\\_GetConnectAttemptDelay](#)

*Set Connection Attempts, Timeout and Delay Parameters in the TIBCO Enterprise Message Service User's Guide*

**tibems\_SetConnectAttemptTimeout**

*Function*

**Purpose**

Modify the connection timeout.

**C Declaration**

```
tibems_status tibems_SetConnectAttemptTimeout(
    tibems_int timeout);
```

**COBOL Call**

```
CALL "tibems_SetConnectAttemptTimeout"
    USING BY VALUE timeout,
           RETURNING tibems-status
END-CALL.
```

**Remarks**

This setting governs all client [tibemsConnection](#) objects. It determines the maximum time (in milliseconds) the client allows to establish a connection to the server.

**Parameters**

Parameter	Description
timeout	Set the connect attempt timeout to this time (in milliseconds). Zero is a special value, which specifies no timeout.  The default value is 0.

**See Also**

[tibems\\_GetConnectAttemptTimeout](#)

*Set Connection Attempts, Timeout and Delay Parameters in the TIBCO Enterprise Message Service User's Guide*

**tibems\_SetExceptionOnFTEvents**

*Function*

**Purpose**

Set whether exception listener is called each step of the fault-tolerant switchover process.

**C Declaration**

```
tibems_status tibems_SetExceptionOnFTEvents(
    tibems_bool* callExceptionListener);
```

## COBOL Call

```
CALL "tibems_SetExceptionOnFTEvents"
  USING BY VALUE callExceptionListener,
  RETURNING tibems-status
END-CALL.
```

### Remarks

This setting determines exception behavior when the fault-tolerant client goes through phases in the failover process. If an exception listener is set on the connection, the callback is invoked when the client detects:

- Disconnection, or TIBEMS\_SERVER\_DISCONNECTED.
- Each reconnect attempt, or TIBEMS\_SERVER\_RECONNECTING.
- Reconnection to the server, or TIBEMS\_SERVER\_RECONNECTED.

This call reports on all events during the failover process. To report only the successful reconnection event, use the [tibems\\_setExceptionOnFTSwitch](#) method instead.

### Parameters

Parameter	Description
callExceptionListener	<p>When true, the connection's <code>ExceptionHandler</code> catches an exception, which contains a status code indicating the state of the fault-tolerant failover: TIBEMS_SERVER_DISCONNECTED, TIBEMS_SERVER_RECONNECTING, or TIBEMS_SERVER_RECONNECTED</p> <p>When false, fault-tolerant failover does not trigger an exception in the client.</p>

### See Also

[tibems\\_GetExceptionOnFTEvents](#)

[tibems\\_setExceptionOnFTSwitch](#)

*Set an Exception Listener in the TIBCO Enterprise Message Service User's Guide*

## tibems\_setExceptionOnFTSwitch

*Function*

### Purpose

Set whether exception listener is called on fault-tolerant switchover.

### C Declaration

```
tibems_status tibems_setExceptionOnFTSwitch(
  tibems_bool callExceptionListener);
```

### COBOL Call

```
CALL "tibems_setExceptionOnFTSwitch"
  USING BY VALUE callExceptionListener,
  RETURNING tibems-status
END-CALL.
```

## Remarks

This setting determines exception behavior when the client successfully switches to a different server (fault-tolerant failover).

## Parameters

Parameter	Description
<code>callExceptionHandler</code>	When true, the connection's <code>ExceptionHandler</code> catches an exception, which contains the name of the new server.  When false, fault-tolerant failover does not trigger an exception in the client.

## See Also

[tibems\\_GetExceptionOnFTSwitch](#)

[tibems\\_GetExceptionOnFTEvents](#)

Set an *Exception Listener* in the *TIBCO Enterprise Message Service User's Guide*

## tibems\_SetMulticastDaemon

*Function*



Along with the multicast feature, this function was deprecated in software release 8.3.0 and will no longer be supported in future releases.

### Purpose

Set the port on which the EMS client will connect to the multicast daemon.

### C Declaration

```
tibems_status tibems_SetMulticastDaemon(
    const char* port);
```

## Parameters

Parameter	Description
<code>port</code>	The multicast daemon port that connections created in this application will connect to.

## Remarks

A connection to the multicast daemon is required when multicast is enabled and a consumer is subscribed to a multicast-enabled topic. Setting the port with this method will override the default port supplied by the server and any port specified in a connection factory for all connections in the application.

This call is not supported in COBOL, and is not supported on z/OS and IBM i systems.

## See Also

[tibemsConnectionFactory\\_SetMulticastDaemon](#)

[tibems\\_SetMulticastEnabled](#)

## tibems\_SetMulticastEnabled

*Function*



Along with the multicast feature, this function was deprecated in software release 8.3.0 and will no longer be supported in future releases.

### Purpose

Set whether message consumers subscribed to multicast-enabled topics in the EMS client application will receive messages over multicast.

### C Declaration

```
tibems_status tibems_SetMulticastEnabled(
    tibems_bool enabled);
```

### Parameters

Parameter	Description
enabled	When true, multicast is enabled for the client. When false, multicast is disabled and the client application will receive all messages over TCP.

### Remarks

When enabled, message consumers subscribed to a multicast-enabled topic will receive messages over multicast. By default, multicast is enabled.

`tibems_SetMulticastEnabled` overrides both the EMS server and factory settings that enable multicast.

This call is not supported in COBOL, and is not supported on z/OS and IBM i systems.

### See Also

[tibemsConnectionFactory\\_SetMulticastDaemon](#)

## tibems\_SetMulticastExceptionHandler

*Function*



Along with the multicast feature, this function was deprecated in software release 8.3.0 and will no longer be supported in future releases.

### Purpose

Set the multicast exception listener for an application.

### C Declaration

```
tibems_status tibems_SetMulticastExceptionHandler(
    tibemsMulticastExceptionHandler listener,
    void* closure);
```

## Parameters

Parameter	Description
listener	Register this multicast exception listener callback.
closure	Receive the closure data in this location, which the program supplied in the call that registered the callback. This is a way to pass application related data to the multicast exception listener callback.

## Remarks

This call is not supported in COBOL.

## See Also

[tibemsMulticastExceptionCallback](#)

## tibems\_SetReconnectAttemptCount

*Function*

## Purpose

Modify the reconnection attempts limit.

## C Declaration

```
tibems_status tibems_SetReconnectAttemptCount(
    tibems_int count);
```

## COBOL Call

```
CALL "tibems_SetReconnectAttemptCount"
    USING BY VALUE count,
    RETURNING tibems-status
END-CALL.
```

## Remarks

This setting governs all client [tibemsConnection](#) objects as they attempt to reconnect to the server after a network disconnect.

## Parameters

Parameter	Description
count	Set the reconnect limit to this value. The default value is 4. The minimum value is 1.

## See Also

[tibems\\_GetReconnectAttemptCount](#)

Set Reconnection Failure Parameters in the *TIBCO Enterprise Message Service User's Guide*.

## tibems\_SetReconnectAttemptDelay

Function

### Purpose

Modify the reconnection delay.

### C Declaration

```
tibems_status tibems_SetReconnectAttemptDelay(
    tibems_int delay);
```

### COBOL Call

```
CALL "tibems_SetReconnectAttemptDelay"
    USING BY VALUE delay,
         RETURNING tibems-status
END-CALL.
```

### Remarks

This setting governs all client [tibemsConnection](#) objects. It determines the delay time between successive attempt to establish a connection to the server. Its value is the time (in milliseconds) between connection attempts.

### Parameters

Parameter	Description
delay	Set the reconnect delay to this value. The default value is 500. The minimum value is 250.

### See Also

[tibems\\_GetReconnectAttemptDelay](#)

*Set Reconnection Failure Parameters* in the *TIBCO Enterprise Message Service User's Guide*.

## tibems\_SetReconnectAttemptTimeout

Function

### Purpose

Modify the reconnection timeout.

### C Declaration

```
tibems_status tibems_SetReconnectAttemptTimeout(
    tibems_int timeout);
```

### COBOL Call

```
CALL "tibems_SetReconnectAttemptTimeout"
    USING BY VALUE timeout,
         RETURNING tibems-status
END-CALL.
```

## Remarks

This setting governs all client `tibemsConnection` objects. It determines the maximum time (in milliseconds) to client will allow to reestablish a connection to the server.

## Parameters

Parameter	Description
<code>timeout</code>	Set the reconnect timeout to this value. Zero is a special value, which specifies no timeout.  The default value is 0.

## See Also

[tibems\\_GetReconnectAttemptTimeout](#)

Set Reconnection Failure Parameters in the TIBCO Enterprise Message Service User's Guide.

## `tibems_SetSocketReceiveBufferSize`

Function

### Purpose

Set the size of socket receive buffers.

### C Declaration

```
tibems_status tibems_SetSocketReceiveBufferSize(
    tibems_int size);
```

### COBOL Call

```
CALL "tibems_SetSocketReceiveBufferSize"
  USING BY VALUE size,
  RETURNING tibems-status
END-CALL.
```

## Remarks

This value overrides the operating system's default for the size of receive buffers associated with sockets that the client uses for connections to the server.

Use this call before creating server connections. This call sets an override buffer size for new socket buffers; it does not change the size of existing socket buffers.

## Parameters

Parameter	Description
<code>size</code>	Sockets use receive buffers of this size (in kilobytes).

## See Also

[tibems\\_GetSocketReceiveBufferSize](#)

## tibems\_SetSocketSendBufferSize

*Function*

### Purpose

Set the size of socket send buffers.

### C Declaration

```
tibems_status tibems_SetSocketSendBufferSize(
    tibems_int size);
```

### COBOL Call

```
CALL "tibems_SetSocketSendBufferSize"
    USING BY VALUE size,
         RETURNING tibems-status
END-CALL.
```

### Remarks

This value overrides the operating system's default for the size of send buffers associated with sockets that the client uses for connections to the server.

Use this call before creating server connections. This call sets an override buffer size for new socket buffers; it does not change the size of existing socket buffers.

### Parameters

Parameter	Description
size	Sockets use send buffers of this size (in kilobytes).

### See Also

[tibems\\_GetSocketSendBufferSize](#)

## tibems\_SetTraceFile

*Function*

### Purpose

Start or stop directing client trace information to a file.

### C Declaration

```
tibems_status tibems_SetTraceFile(
    const char* fileName);
```

### COBOL Call

```
CALL "tibems_SetTraceFile"
    USING BY REFERENCE fileName,
         RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
<code>fileName</code>	The name of the file to which tracing information should be directed, or NULL to stop directing tracing to a file.

## Remarks

This call directs client tracing information to the file specified by the `fileName` argument. This function does not generate tracing; tracing must be enabled separately. Tracing can be enabled using the `set server client_trace` command in the administration tool, or, for SSL tracing, using the [tibemsSSL\\_SetTrace](#) or `tibemsSSL_SetDebugTrace` function.

See the *TIBCO Enterprise Message Service User's Guide* for more information about the administration tool.

To stop sending tracing information to the file and resume regular tracing to `stderr` or `stdout`, call `tibems_SetTraceFile`, passing NULL for `fileName` parameter.

Status Code	Description
<code>TIBEMS_ILLEGAL_STATE</code>	Tracing information is already being directed to a file, and the <code>fileName</code> argument was not NULL.
<code>TIBEMS_IO_FAILED</code>	The file specified by <code>fileName</code> could not be opened in append mode.

## `tibems_Sleep`

*Function*

### Purpose

Sleep thread.

### C Declaration

```
void tibems_Sleep(
    tibems_long milliseconds);
```

### COBOL Call

```
CALL "tibems_Sleep"
  USING BY VALUE milliseconds
END-CALL.
```



COBOL usage of milliseconds is COMP-2.

## Parameters

Parameter	Description
<code>milliseconds</code>	Sleep for this interval (in milliseconds).

**Remarks**

This call instructs its calling thread to sleep for a fixed interval (in milliseconds).

**tibems\_Version**

*Function*

**Purpose**

Return the EMS library version number.

**C Declaration**

```
const char* tibems_Version(void);
```

**COBOL Call**

```
CALL "tibems_Version"  
  RETURNING tibems-Pointer  
  END-CALL.
```

tibems-Pointer has usage pointer.



After this COBOL call, store the tibems-Pointer data item to tibems-Cobol-Char, using the following SET statement:

```
SET ADDRESS OF tibems-Cobol-Char TO tibems-Pointer.
```

**Remarks**

This string represents the three-part version number of the release (*major.minor.update*).

# Administration

This section documents the C Admin API functions used to query the runtime state of EMS messaging applications.

## Administration Overview

TIBCO Enterprise Message Service C Admin API provides a C program with the ability to query the tibemsd server and obtain information regarding the state of the tibemsd, message consumers, producers, topics and queues at runtime. TIBCO Enterprise Message Service C Admin API closely mimics a subset of the Java Admin API query functionality. It does not provide configuration capability.

### Including the Administration Library

The C Admin API is provided as a separate library. For information about including the C Admin API, see the *C Programmer's Checklist* of the *TIBCO Enterprise Message Service User's Guide*.

## tibemsAdmin

*Type*

### Purpose

Represent an administrative connection to the server.

### Remarks

`tibemsAdmin` provides an administrative connection to the server. To use these functions, first create an administrative connection to the server with `tibemsAdmin_Create`. With that connection you can retrieve information about the server and its components at runtime.

Function	Description
<code>tibemsAdmin_Close</code>	Close the administrative connection to the server.
<code>tibemsAdmin_Create</code>	Create an administration connection to a server.
<code>tibemsAdmin_GetCommandTimeout</code>	Get the command timeout.
<code>tibemsAdmin_GetConsumer</code>	Get consumer with specified ID.
<code>tibemsAdmin_GetConsumers</code>	Return consumers matching specified filters.
<code>tibemsAdmin_GetInfo</code>	Get the current set of server metrics.
<code>tibemsAdmin_GetProducerStatistics</code>	Return statistical information about producers that match the specified parameters.
<code>tibemsAdmin_GetQueue</code>	Get information about a queue of the given queue name.
<code>tibemsAdmin_GetQueues</code>	Get the queues that match the given pattern and the given permanence type.
<code>tibemsAdmin_GetSubscriptions</code>	Return subscriptions matching the specified filters.

Function	Description
<a href="#">tibemsAdmin_GetTopic</a>	Get the topic for the given topic name.
<a href="#">tibemsAdmin_GetTopics</a>	Get the topics that match the given pattern and the given permanence type.
<a href="#">tibemsAdmin_SetCommandTimeout</a>	Set the command timeout.
<a href="#">tibemsAdmin_SetExceptionHandler</a>	Set an exception listener for the connection used by the administration API to communicate with the EMS server.

### Related Types

[tibemsCollection](#)

[tibemsConsumerInfo](#)

[tibemsProducerInfo](#)

[tibemsQueueInfo](#)

[tibemsTopicInfo](#)

## tibemsAdmin\_Close

*Function*

### Purpose

Close the administrative connection to the server.

### C Declaration

```
tibems_status tibemsAdmin_Close(
    tibemsAdmin admin);
```

### COBOL Call

```
CALL "tibemsAdmin_Close"
  USING BY VALUE admin,
  RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
admin	The administrative connection to be closed.

### See Also

[tibemsAdmin\\_Create](#)

## tibemsAdmin\_Create

Function

### Purpose

Create an administration connection to a server.

### C Declaration

```
tibems_status tibemsAdmin_Create(
    tibemsAdmin* admin,
    const char* url,
    const char* userName,
    const char* password,
    tibemsSSLParams sslparams);
```

### COBOL Call

```
CALL "tibemsAdmin_Create"
  USING BY REFERENCE admin,
        BY REFERENCE url,
        BY REFERENCE userName,
        BY REFERENCE password,
        BY VALUE sslParams,
        RETURNING tibems-status
END-CALL.
```



sslParams has usage pointer.

### Parameters

Parameter	Description
admin	Store the new administrative connection in this location.
url	Find the EMS server at this URL.
userName	Authenticate the client program to the server with this username.
password	Authenticate the client program to the server with this password.
sslparams	Establish SSL communication using these parameters. See <i>SSL server parameters</i> in the <i>TIBCO Enterprise Message Service User's Guide</i> for more information.

### Status Codes

Status Code	Description
<a href="#">TIBEMS_SERVER_NOT_CONNECTED</a>	<ul style="list-style-type: none"> <li>No server is running at the specified URL.</li> <li>The call could not communicate with a server because of mismatched SSL and TCP protocols.</li> <li>Other error situations are possible.</li> </ul>

Status Code	Description
<a href="#">TIBEMS_SECURITY_EXCEPTION</a>	<ul style="list-style-type: none"> <li>The server rejected the connection because the username or password was invalid.</li> <li>SSL setup is incorrect.</li> </ul>

## tibemsAdmin\_GetCommandTimeout

*Function*

### Purpose

Get the command timeout.

### C Declaration

```
tibems_status tibemsAdmin_GetCommandTimeout(
    tibemsAdmin admin,
    tibems_long* timeout);
```

### COBOL Call

```
CALL "tibemsAdmin_GetCommandTimeout"
    USING BY VALUE admin,
         BY REFERENCE timeout,
         RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
admin	Get the command timeout of this administrative connection.
timeout	The function stores the timeout in this location.

### Remarks

Get the command timeout in milliseconds. The command timeout determines how long to wait for the server to respond to a command. If the server does not respond within the timeout limit, the command throws an exception. The default timeout is 60000 (60 seconds).

### See Also

[tibemsAdmin\\_SetCommandTimeout](#)

## tibemsAdmin\_GetConsumer

*Function*

### Purpose

Get consumer with specified ID.

### C Declaration

```
tibems_status tibemsAdmin_GetConsumer(
    tibemsAdmin admin,
```

```
tibemsConsumerInfo* consumerInfo,
tibems_long ID);
```

### COBOL Call

```
CALL "tibemsAdmin_GetConsumer"
  USING BY VALUE      admin,
         BY REFERENCE consumerInfo,
         BY VALUE      consumerID,
         RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
admin	Get a consumer using this administrative connection.
consumerInfo	The functions stores information about the consumer in this location.
ID	The consumer ID for which the function will retrieve information.

### Remarks

Return the consumer object with specified ID. The returned consumer object contains information about consumer known to server, including details, available statistics. If a consumer with the specified ID does not exist, the function returns NULL.

### Status Codes

Status Code	Description
TIBEMS_TIMEOUT	The administrative query timed out while waiting for a server response.
TIBEMS_NOT_FOUND	No consumer found matching the requested ID.

### See Also

[tibemsConsumerInfo](#)

## tibemsAdmin\_GetConsumers

*Function*

### Purpose

Return consumers matching specified filters.

### C Declaration

```
tibems_status tibemsAdmin_GetConsumers(
  tibemsAdmin admin,
  tibemsCollection* collection,
  tibems_long connectionID,
  const char* username,
  tibemsDestinationInfo destination,
  tibems_bool durable,
  tibems_int dataFlags)
```

## COBOL Call

```
CALL "tibemsAdmin_GetConsumers"
  USING BY VALUE admin,
        BY REFERENCE collection,
        BY VALUE connectionID,
        BY REFERENCE username,
        BY VALUE destination,
        BY VALUE durable,
        BY VALUE dataFlags,
        RETURNING tibems-status
END-CALL.
```



collection has usage pointer.

### Parameters

Parameter	Description
admin	Get consumers using this administrative connection.
collection	Store the returned consumer data in the location specified here, as a collection of <code>tibemsConsumerInfo</code> objects.
connectionID	connectionID is reserved for future use and must be set to zero.
username	If specified, only consumers for connections that use the specified user name will be returned. Specify NULL if all consumers should be returned.
destination	<p>If specified, only consumers on destinations of the same type and matching this destination name will be returned. destination can be:</p> <ul style="list-style-type: none"> <li>• A <code>tibemsQueueInfo</code> or <code>tibemsTopicInfo</code> object.</li> <li>• <code>TIBEMS_INVALID_ADMIN_ID</code> to return all consumers.</li> </ul> <p>The destination name may include wildcards.</p>

Parameter	Description
durable	<p>(If TRUE) Specify that only durable topic subscribers should be returned.</p> <p>Applies only to topic subscribers, and when included prevents the function from returning non-durable topic consumers. However, it does not affect which queue consumers are returned.</p>
dataFlags	<p>Specify what information will be returned for each consumer that matches the filter criteria. Possible values for this parameter are:</p> <ul style="list-style-type: none"> <li>• GET_STAT—gets a <code>tibemsStatData</code> for each consumer.</li> <li>• GET_DETAILED_STAT—gets a <code>tibemsCollection</code> of <code>tibemsDetailedDestStat</code> objects for each consumer</li> </ul> <p>When no flag is specified, the returned information does not include statistics.</p> <p>If statistics are disabled in the server, no statistics will be returned regardless of the flag specified in this parameter.</p>

### Remarks

Return a list of consumers matching the specified filters. The consumers are returned in a `tibemsCollection`; if no consumers matching the filter criteria exist in the server, then no `tibemsCollection` will be returned.

The returned consumers are not sorted and are placed in the `tibemsCollection` object in any order. Your application may need to sort the consumers into a specific order if required.

### Example 1

For example, this call returns all consumers known to the server, but does not include statistical information for each consumer:

```
tibemsAdmin      admin;
tibemsCollection consumerInfoCollection;
tibems_status    status;

status = tibemsAdmin_GetConsumers(admin, &consumerInfoCollection, 0L, NULL,
TIBEMS_ADMIN_INVALID_ID, TIBEMS_FALSE, 0);
```

### Example 2

This call returns all queue consumers and all durable topic consumers:

```
tibemsAdmin      admin;
tibemsCollection consumerInfoCollection;
tibems_status    status;
```

```
status = tibemsAdmin_GetConsumers(admin, &consumerInfoCollection, 0L, NULL,
TIBEMS_ADMIN_INVALID_ID, TIBEMS_TRUE, 0);
```

### Example 3

This call returns all durable topic consumers that subscribe to any topic matching topic news.\*. If statistics are enabled in the server, the returned `tibemsConsumerInfo` objects will include detailed statistics about the consumers.

```
tibemsAdmin      admin;
tibemsCollection consumerInfoCollection;
tibems_status    status;
tibemsTopicInfo  topicInfo;

status = tibemsTopicInfoCreate(&topicInfo, "news.*");
status = tibemsAdmin_GetConsumers(admin, &consumerInfoCollection, 0L, NULL,
topicInfo, TIBEMS_TRUE, TIBEMS_GET_DETAILED_STAT);
```

### Example 4

This call returns all queue consumers created by user `OrderProcessor` and receiving messages from all queues matching name `purchase.order.>`. Each `tibemsConsumerInfo` object will include the full statistics available for the consumer.

```
tibemsAdmin      admin;
tibemsCollection consumerInfoCollection;
tibems_status    status;
tibemsQueueInfo  queueInfo;

status = tibemsQueueInfoCreate(&queueInfo, "purchase.order.>");
status = tibemsAdmin_GetConsumers(admin, &consumerInfoCollection, 0L,
"OrderProcessor", queueInfo, TIBEMS_FALSE, TIBEMS_GET_DETAILED_STAT);
```

### Status Codes

Status Code	Description
TIBEMS_TIMEOUT	The administrative query timed out while waiting for a server response.
TIBEMS_NOT_FOUND	No consumers found matching the specified filters.

### See Also

[tibemsStatData](#)

[tibemsDetailedDestStat](#)

## tibemsAdmin\_GetInfo

*Function*

### Purpose

Get the current set of server metrics.

### C Declaration

```
tibems_status tibemsAdmin_GetInfo(
    tibemsAdmin admin,
    tibemsServerInfo* serverInfo);
```

### COBOL Call

```
CALL "tibemsAdmin_GetInfo"
  USING BY VALUE      admin,
        BY REFERENCE serverInfo,
```

```
RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
admin	Get server metrics using this administrative connection.
serverInfo	The function stores the server metrics in this location.

### Status Codes

Status Code	Description
TIBEMS_TIMEOUT	The administrative query timed out while waiting for a server response.

## tibemsAdmin\_GetProducerStatistics

*Function*

### Purpose

Return statistical information about producers that match the specified parameters.

### C Declaration

```
tibems_status tibemsAdmin_GetProducerStatistics(
    tibemsAdmin admin,
    tibemsCollection* collection,
    tibems_long connectionID,
    const char* username,
    tibemsDestinationInfo destination);
```

### COBOL Call

```
CALL "tibemsAdmin_GetProducerStatistics"
  USING BY VALUE admin,
        BY REFERENCE prodInfos,
        BY REFERENCE username,
        BY VALUE connectionID,
        BY VALUE destination,
        RETURNING tibems-status
END-CALL.
```



prodInfos has usage pointer.

### Parameters

Parameter	Description
admin	Get the producer statistics using this administrative connection.
collection	Store the returned producer data in the location specified here, as a collection of tibemsProducerInfo objects.
connectionID	Must be set to zero. This parameter is reserved for future use.

Parameter	Description
username	If specified, only producers for connections that use the specified user name will be returned. Specify NULL if all producers should be returned.
destination	If specified, only producers on destinations of the same type and matching this destination name will be returned. destination can be: <ul style="list-style-type: none"> <li>• A <a href="#">tibemsQueueInfo</a> or <a href="#">tibemsTopicInfo</a> object.</li> <li>• TIBEMS_INVALID_ADMIN_ID to return all producers.</li> </ul> The destination name may include wildcards.

### Remarks

Return information about message producers, including the statistical information about producers with specified parameters.

### Status Codes

Status Code	Description
TIBEMS_TIMEOUT	The administrative query timed out while waiting for a server response.
TIBEMS_NOT_FOUND	No producers found matching the specified parameters.

### See Also

[tibemsStatData](#)

[tibemsDetailedDestStat](#)

## tibemsAdmin\_GetQueue

*Function*

### Purpose

Get information about a queue of the given queue name.

### C Declaration

```
tibems_status tibemsAdmin_GetQueue(
    tibemsAdmin admin,
    tibemsQueueInfo* queueInfo,
    const char* queueName);
```

### COBOL Call

```
CALL "tibemsAdmin_GetQueue"
  USING BY VALUE admin,
        BY REFERENCE queueInfo,
        BY REFERENCE name,
        RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
admin	Get information about a queue using this administrative connection.
queueInfo	Store information about the specified queue in this location.
queueName	Name of the queue for which information will be retrieved.

## Remarks

Get a `tibemsQueueInfo` object for the specified queue name. If the `queueName` does not exist, the function returns `TIBEMS_NOT_FOUND`.

## Status Codes

Status Code	Description
TIBEMS_TIMEOUT	The administrative query timed out while waiting for a server response.
TIBEMS_NOT_FOUND	No queue found matching the specified <code>queueName</code> .

## See Also

[tibemsQueueInfo](#)

## tibemsAdmin\_GetQueues

*Function*

### Purpose

Get the queues that match the given pattern and the given permanence type.

### C Declaration

```
tibems_status tibemsAdmin_GetQueues(
    tibemsAdmin admin,
    tibemsCollection* collection,
    const char* pattern,
    tibems_int permType);
```

### COBOL Call

```
CALL "tibemsAdmin_GetQueues"
  USING BY VALUE admin,
        BY REFERENCE collection,
        BY REFERENCE pattern,
        BY VALUE permType,
        RETURNING tibems-status
END-CALL.
```



collection has usage pointer.

## Parameters

Parameter	Description
admin	Get information about queues using this administration connection.
collection	The function stores the returned queue data in the location specified here, as a collection of <code>tibemsQueueInfo</code> objects.
pattern	The queue name pattern that must be matched.  The pattern may contain the wildcards <code>*</code> and <code>&gt;</code> . A pattern of <code>&gt;</code> or <code>NULL</code> will return all queues that exist in the server.  For more information, see <i>Wildcards in Queues</i> in the <i>TIBCO Enterprise Message Service User's Guide</i> .
permType	The permanence type of the queue must match the type given here. Possible permanence types are: <ul style="list-style-type: none"> <li>• <code>TIBEMS_DEST_GET_ALL</code> – Return all queues that match the pattern.</li> <li>• <code>TIBEMS_DEST_GET_STATIC</code> – Return only static queues that match the pattern.</li> <li>• <code>TIBEMS_DEST_GET_DYNAMIC</code> – Return only dynamic queues that match the pattern.</li> <li>• <code>TIBEMS_DEST_GET_NOTEMP</code> – Do not return any temporary queues.</li> </ul> A <code>NULL</code> value matches all queues.

## Status Codes

Status Code	Description
<code>TIBEMS_TIMEOUT</code>	The administrative query timed out while waiting for a server response.
<code>TIBEMS_NOT_FOUND</code>	No queues found matching the specified pattern and performance type.

## See Also

[tibemsQueueInfo](#)

## tibemsAdmin\_GetSubscriptions

*Function*

### Purpose

Return subscriptions matching the specified filters.

### C Declaration

```
tibems_status tibemsAdmin_GetSubscriptions(
    tibemsAdmin admin,
    tibemsCollection* collection,
    tibems_int filterFlags,
    const char* name,
    const char* topicName);
```

## COBOL Call

```
CALL "tibemsAdmin_GetSubscriptions"
  USING BY VALUE admin,
        BY REFERENCE collection,
        BY VALUE filterFlags,
        BY REFERENCE name,
        BY REFERENCE topicName,
        RETURNING tibems-status
END-CALL.
```



collection has usage pointer.

## Parameters

Parameter	Description
admin	Get information about subscriptions using this administration connection.
collection	Store the returned subscription data in the location specified here, as a collection of tibemsSubscriptionInfo objects.
filterFlags	Value can be any combination of the flags: <ul style="list-style-type: none"> <li>TIBEMS_SUBSCRIPTIONS_FILTER_DURABLE_ONLY</li> <li>TIBEMS_SUBSCRIPTIONS_FILTER_NO_DURABLE</li> <li>TIBEMS_SUBSCRIPTIONS_FILTER_SHARED_ONLY</li> <li>TIBEMS_SUBSCRIPTIONS_FILTER_NO_SHARED</li> </ul>
name	Specifies that only subscriptions with this name should be returned.
topicName	Specifies that only subscriptions on this topic name should be returned.

## Remarks

Return a list of all subscriptions that match the specified filters. The subscriptions are returned in a `tibemsCollection`; if no subscription matching the filter criteria exist in the server, then no `tibemsCollection` is returned.

The returned subscriptions are not sorted and are placed in the `tibemsCollection` object in any order. Your application may need to sort the subscriptions into a specific order.

For example, the following returns all shared subscriptions known to the server:

```
tibemsAdmin      admin;
tibemsCollection subscriptionsInfo;
tibems_status    status;

status = tibemsAdim_GetSubscriptions(admin, &subscriptionsInfo, 0, NULL, NULL);
```

The following returns all durable (shared or not shared) subscriptions known to the server:

```
tibemsAdmin      admin;
tibemsCollection subscriptionsInfo;
tibems_status    status;

status = tibemsAdim_GetSubscriptions(admin, &subscriptionsInfo,
TIBEMS_SUBSCRIPTIONS_FILTER_DURABLE_ONLY, NULL, NULL);
```

The following returns all shared durable subscriptions on any topic matching topic news.\*:

```
tibemsAdmin      admin;
tibemsCollection subscriptionsInfo;
tibems_status    status;

status = tibemsAdmin_GetSubscriptions(admin, &subscriptionsInfo,
TIBEMS_SUBSCRIPTIONS_FILTER_DURABLE_ONLY + TIBEMS_SUBSCRIPTIONS_FILTER_SHARED_ONLY,
NULL, "news.*");
```

### Status Codes

Status Code	Description
TIBEMS_TIMEOUT	The administrative query timed out while waiting for a server response.
TIBEMS_NOT_FOUND	No subscription found matching the specified filters.

## tibemsAdmin\_GetTopic

*Function*

### Purpose

Get the topic for the given topic name.

### C Declaration

```
tibems_status tibemsAdmin_GetTopic(
    tibemsAdmin admin,
    tibemsTopicInfo* topicInfo,
    const char* topicName);
```

### COBOL Call

```
CALL "tibemsAdmin_GetTopic"
    USING BY VALUE admin,
         BY REFERENCE topicInfo,
         BY REFERENCE name,
         RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
admin	Get information about a topic using this administrative connection.
topicInfo	Stores information about the topic in this location.
topicName	The name of the topic for which information will be retrieved.

### Remarks

This function gets a `tibemsTopicInfo` object for the specified topic name. If the `topicName` does not exist, the function returns `TIBEMS_NOT_FOUND`.

## Status Codes

Status Code	Description
TIBEMS_TIMEOUT	The administrative query timed out while waiting for a server response.
TIBEMS_NOT_FOUND	No topic found matching the specified topicName.

## See Also

[tibemsTopicInfo](#)

## tibemsAdmin\_GetTopics

*Function*

### Purpose

Get the topics that match the given pattern and the given permanence type.

### C Declaration

```
tibems_status tibemsAdmin_GetTopics(
    tibemsAdmin admin,
    tibemsCollection* collection,
    const char* pattern,
    tibems_int permType);
```

### COBOL Call

```
CALL "tibemsAdmin_GetTopics"
  USING BY VALUE admin,
        BY REFERENCE collection,
        BY REFERENCE pattern,
        BY VALUE permType,
        RETURNING tibems-status
END-CALL.
```



collection has usage pointer.

### Parameters

Parameter	Description
admin	Get information about topics using this administrative connection.
collection	Store the returned topic data in the location specified here, as a collection of tibemsTopicInfo objects.
pattern	The topic name pattern that must be matched. The pattern may contain the wildcards * and >. A pattern of > or NULL will return all topics. See the <i>TIBCO Enterprise Message Service User's Guide</i> for information about working with wildcards in topics.

Parameter	Description
permType	<p>The permanence type of the topic must match the type given here. Possible permanence types are:</p> <ul style="list-style-type: none"> <li>TIBEMS_DEST_GET_ALL – Return all topics that match the pattern.</li> <li>TIBEMS_DEST_GET_STATIC – Return only static topics that match the pattern.</li> <li>TIBEMS_DEST_GET_DYNAMIC – Return only dynamic topics that match the pattern.</li> <li>TIBEMS_DEST_GET_NOTEMP – Do not return any temporary topics.</li> </ul> <p>A NULL value matches all topics.</p>

### Status Codes

Status Code	Description
TIBEMS_TIMEOUT	The administrative query timed out while waiting for a server response.
TIBEMS_NOT_FOUND	No topics found matching the specified pattern and performance type.

### See Also

[tibemsTopicInfo](#)

## tibemsAdmin\_SetCommandTimeout

*Function*

### Purpose

Set the command timeout.

### C Declaration

```
tibems_status tibemsAdmin_SetCommandTimeout(
    tibemsAdmin admin,
    tibems_long timeout);
```

### COBOL Call

```
CALL "tibemsAdmin_SetCommandTimeout"
  USING BY VALUE admin,
        BY VALUE timeout,
        RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
admin	Set the command timeout on this administrative connection.

Parameter	Description
timeout	The length of time, in milliseconds, to wait for the server to respond to a command. When not specified, the default command timeout is 60000 (60 seconds).

### Remarks

Set the command timeout. The command timeout determines how long, in milliseconds, the command waits for the server to respond. If the server does not respond within the timeout limit, the command throws an exception. The default timeout is 60000 (60 seconds).

### See Also

[tibemsAdmin\\_GetCommandTimeout](#)

## tibemsAdmin\_SetExceptionListener

*Function*

### Purpose

Set an exception listener for the connection used by the administration API to communicate with the EMS server.

### C Declaration

```
tibems_status tibemsAdmin_SetExceptionListener(
    tibemsAdmin admin,
    tibemsExceptionCallback listener,
    const void* closure);
```

### Parameters

Parameter	Description
admin	Register the exception listener callback on this administration connection.
listener	Register this exception listener callback. See the <i>TIBCO Enterprise Message Service User's Guide</i> .
closure	Register this closure argument.

### Remarks

This is an alternate pathway for alerting a client program of connection problems. The program defines an exception listener callback function, and calls this function to register the callback and a closure argument. When the client library detects a connection problem, it calls the callback with a status code that identifies the problem.

This call is not supported in COBOL.

### See Also

[tibemsConnection\\_SetExceptionListener](#)

[Asynchronous Errors](#).

[tibemsExceptionCallback](#)

## tibemsCollection

*Type*

### Purpose

Represent a collection of administrative objects.

### Remarks

Collections provide an array-like set of administrative objects for use by the calling program. Administrative objects include [tibemsQueueInfo](#), [tibemsTopicInfo](#), [tibemsConsumerInfo](#), [tibemsProducerInfo](#), and [tibemsDetailedDestStat](#) objects.



Collections are not thread-safe and should only be operated on by a single application thread.

Function	Description
<a href="#">tibemsCollection_Destroy</a>	Destroy a collection.
<a href="#">tibemsCollection_GetCount</a>	Get the count of objects in a collection.
<a href="#">tibemsCollection_GetFirst</a>	Get the first object in a collection.
<a href="#">tibemsCollection_GetNext</a>	Get the next object in a collection.

### See Also

[tibemsAdmin\\_GetQueues](#)

[tibemsAdmin\\_GetTopics](#)

[tibemsAdmin\\_GetConsumers](#)

[tibemsAdmin\\_GetProducerStatistics](#)

[tibemsConsumerInfo\\_GetDetailedStatistics](#)

[tibemsProducerInfo\\_GetDetailedStatistics](#)

## tibemsCollection\_Destroy

*Function*

### Purpose

Destroy a collection.

### C Declaration

```
tibems_status tibemsCollection_Destroy(
    tibemsCollection collection);
```

### COBOL Call

```
CALL "tibemsCollection_Destroy"
  USING BY VALUE collection,
  RETURNING tibems-status
END-CALL.
```



collection has usage pointer.

**Parameters**

Parameter	Description
collection	The collection object to be destroyed.

**Remarks**

Collections are created internally by the EMS library. Use `tibemsCollection_Destroy` to remove collections of `tibemsQueueInfo`, `tibemsTopicInfo`, `tibemsConsumerInfo`, and `tibemsProducerInfo` objects. Collections of `tibemsDetailedDestStat` objects cannot be directly destroyed by the user application. Attempting to destroy a `tibemsDetailedDestStat` object returns `TIBEMS_NOT_PERMITTED`.

**Status Codes**

Status Code	Description
TIBEMS_NOT_PERMITTED	The collection may not be directly destroyed by the application.

**tibemsCollection\_GetCount**

*Function*

**Purpose**

Get the count of objects in a collection.

**C Declaration**

```
tibems_status tibemsCollection_GetCount(
    tibemsCollection collection,
    tibems_int* count);
```

**COBOL Call**

```
CALL "tibemsCollection_GetCount"
  USING BY VALUE collection,
        BY REFERENCE count,
        RETURNING tibems-status
END-CALL.
```



collection has usage pointer.

**Parameters**

Parameter	Description
collection	Get the number of objects in this collection.
count	The function stores the count of objects in the collection in this location.

## tibemsCollection\_GetFirst

*Function*

### Purpose

Get the first object in a collection.

### C Declaration

```
tibems_status tibemsCollection_GetFirst(
    tibemsCollection collection,
    collectionObj* object);
```

### COBOL Call

```
CALL "tibemsCollection_GetFirst"
    USING BY VALUE collection,
         BY REFERENCE obj,
         RETURNING tibems-status
END-CALL.
```



collection has usage pointer.

### Parameters

Parameter	Description
collection	Get the first object in this collection.
object	The function stores the first object in the collection in this location.

### Remarks

This function must be called before any calls to `tibemsCollection_GetNext`.

## tibemsCollection\_GetNext

*Function*

### Purpose

Get the next object in a collection.

### C Declaration

```
tibems_status tibemsCollection_GetNext(
    tibemsCollection collection,
    collectionObj* object);
```

### COBOL Call

```
CALL "tibemsCollection_GetNext"
    USING BY VALUE collection,
         BY REFERENCE obj,
         RETURNING tibems-status
END-CALL.
```



collection has usage pointer.

## Parameters

Parameter	Description
collection	Get the next object in this collection.
object	The function stores the next object in the collection in this location.

## Remarks

This function gets the next object in a collection. If there are no more objects in the collection, `tibemsCollection_GetNext` returns `TIBEMS_NOT_FOUND`.

The function `tibemsCollection_GetFirst` must be called before `tibemsCollection_GetNext` is called for the first time.

## Status Codes

Status Code	Description
<code>TIBEMS_NOT_INITIALIZED</code>	The collection has not been initialized with a call to <code>tibemsCollection_GetFirst</code> .
<code>TIBEMS_NOT_FOUND</code>	There are no more objects in the collection.

## tibemsConsumerInfo

*Type*

### Purpose

Represents a message consumer in the server.

Function	Description
<code>tibemsConsumerInfo_Destroy</code>	Destroy a <code>consumerInfo</code> object.
<code>tibemsConsumerInfo_GetCreateTime</code>	Get a consumer's creation time in milliseconds.
<code>tibemsConsumerInfo_GetCurrentMsgCountSentByServer</code>	Get the number of messages sent to consumer and not yet acknowledged by consumer's session.
<code>tibemsConsumerInfo_GetCurrentMsgSizeSentByServer</code>	Get the combined size of messages sent to consumer and not yet acknowledged by consumer's session.
<code>tibemsConsumerInfo_GetDestinationName</code>	Get the consumer's destination name.

Function	Description
<code>tibemsConsumerInfo_GetDestinationType</code>	Get consumer's destination type.
<code>tibemsConsumerInfo_GetDetailedStatistics</code>	Get detailed statistics for a wildcarded consumer.
<code>tibemsConsumerInfo_GetDurableName</code>	Get the name of the consumer's durable subscription.
<code>tibemsConsumerInfo_GetElapsedSinceLastAcknowledged</code>	Get the approximate number of milliseconds elapsed since the last time a message sent to this consumer was acknowledged by the consumer's session.
<code>tibemsConsumerInfo_GetElapsedSinceLastSent</code>	Get the approximate number of milliseconds elapsed since last time the server sent a message to this consumer.
<code>tibemsConsumerInfo_GetID</code>	Get the consumer's unique ID.
<code>tibemsConsumerInfo_GetPendingMessageCount</code>	Get the number of pending messages for a topic consumer.
<code>tibemsConsumerInfo_GetSharedSubscriptionName</code>	Get the shared subscription name for a consumer.
<code>tibemsConsumerInfo_GetPendingMessageSize</code>	Get the combined size of pending messages for a topic consumer.
<code>tibemsConsumerInfo_GetStatistics</code>	Get total statistics for a consumer.
<code>tibemsConsumerInfo_GetTotalAcknowledgedCount</code>	Get the total number of messages which were delivered to this consumer and have been acknowledged by the consumer's session.
<code>tibemsConsumerInfo_GetTotalMsgCountSentByServer</code>	Get the total number of messages the server sent to this consumer since the consumer was created.

Function	Description
<code>tibemsConsumerInfo_IsActive</code>	Get the active status of the consumer.
<code>tibemsConsumerInfo_IsConnected</code>	Get the connection status of the consumer.
<code>tibemsConsumerInfo_IsConnectionConsumer</code>	Get whether this is connection consumer.
<code>tibemsConsumerInfo_IsShared</code>	Get the shared subscription status of the consumer.

### Related Types

`tibemsStatData`

`tibemsDetailedDestStat`

## `tibemsConsumerInfo_Destroy`

*Function*

### Purpose

Destroy a `consumerInfo` object.

### C Declaration

```
tibems_status tibemsConsumerInfo_Destroy(
    tibemsConsumerInfo consumerInfo);
```

### COBOL Call

```
CALL "tibemsConsumerInfo_Destroy"
  USING BY VALUE consumerInfo,
  RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
<code>consumerInfo</code>	<code>consumerInfo</code> object to be destroyed.

## `tibemsConsumerInfo_GetCreateTime`

*Function*

### Purpose

Get a consumer's creation time in milliseconds.

### C Declaration

```
tibems_status tibemsConsumerInfo_GetCreateTime(
    tibemsConsumerInfo consumerInfo,
    tibems_long* ctime);
```

**COBOL Call**

```
CALL "tibemsConsumerInfo_GetCreateTime"
  USING BY VALUE consumerInfo,
        BY REFERENCE created,
        RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
consumerInfo	Get the creation time of this consumer.
ctime	The function stores the create time in this location.

**tibemsConsumerInfo\_GetCurrentMsgCountSentByServer***Function***Purpose**

Get the number of messages sent to consumer and not yet acknowledged by consumer's session.

**C Declaration**

```
tibems_status tibemsConsumerInfo_GetCurrentMsgCountSentByServer(
    tibemsConsumerInfo consumerInfo,
    tibems_long* count);
```

**COBOL Call**

```
CALL "tibemsConsumerInfo_GetCurrentMsgCountSentByServer"
  USING BY VALUE consumerInfo,
        BY REFERENCE count,
        RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
consumerInfo	Get the number of messages sent to this consumer.
count	The function stores the number of messages in this location.

**Remarks**

This function gets the number of messages sent to but not yet acknowledged by the consumer. For topic consumers, this number is included in the number of pending messages returned by `tibemsConsumerInfo_GetPendingMessageCount`.

**See Also**

[tibemsConsumerInfo\\_GetPendingMessageCount](#)

## tibemsConsumerInfo\_GetCurrentMsgSizeSentByServer

*Function*

### Purpose

Get the combined size of messages sent to consumer and not yet acknowledged by consumer's session.

### C Declaration

```
tibems_status tibemsConsumerInfo_GetCurrentMsgSizeSentByServer(
    tibemsConsumerInfo consumerInfo,
    tibems_long* size);
```

### COBOL Call

```
CALL "tibemsConsumerInfo_GetCurrentMsgSizeSentByServer"
    USING BY VALUE consumerInfo,
         BY REFERENCE size,
         RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
consumerInfo	Get the unacknowledged message size for this consumer.
size	The function stores the total size of sent messages in this location.

### Remarks

This function gets the combined size of messages sent to consumer and not yet acknowledged by consumer's session. For topic consumers this size is included into the combined size of pending messages returned by `tibemsConsumerInfo_GetPendingMessageSize`.

### See Also

[tibemsConsumerInfo\\_GetPendingMessageSize](#)

## tibemsConsumerInfo\_GetDestinationName

*Function*

### Purpose

Get the consumer's destination name.

### C Declaration

```
tibems_status tibemsConsumerInfo_GetDestinationName(
    tibemsConsumerInfo consumerInfo,
    char* name,
    tibems_int name_len);
```

### COBOL Call

```
CALL "tibemsConsumerInfo_GetDestinationName"
    USING BY VALUE consumerInfo,
         BY REFERENCE destName,
         BY VALUE name_len,
```

```
RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
consumerInfo	Get the destination name of this consumer.
name	The function stores the destination name in this location.
name_len	Length of the name buffer.

## tibemsConsumerInfo\_GetDestinationType

*Function*

### Purpose

Get consumer's destination type.

### C Declaration

```
tibems_status tibemsConsumerInfo_GetDestinationType(
    tibemsConsumerInfo consumerInfo,
    tibemsDestinationType* type);
```

### COBOL Call

```
CALL "tibemsConsumerInfo_GetDestinationType"
    USING BY VALUE consumerInfo,
         BY REFERENCE destType,
         RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
consumerInfo	Get the destination type of this consumer.
type	The function stores the destination type in this location.

### See Also

[tibemsDestinationType](#)

## tibemsConsumerInfo\_GetDetailedStatistics

*Function*

### Purpose

Get detailed statistics for a wildcarded consumer.

### C Declaration

```
tibems_status tibemsConsumerInfo_GetDetailedStatistics(
    tibemsConsumerInfo consumerInfo,
    tibemsCollection* collection);
```

## COBOL Call

```
CALL "tibemsConsumerInfo_GetDetailedStatistics"
  USING BY VALUE consumerInfo,
        BY REFERENCE collection,
        RETURNING tibems-status
END-CALL.
```



collection has usage pointer.

## Parameters

Parameter	Description
consumerInfo	Get statistics for this consumer.
collection	Store the collection of tibemsDetailedDestStat objects in this location.

## Remarks

Return detailed statistics for the consumer, giving a breakdown of the consumer's aggregate statistics across all destinations that it has received messages on.

This function returns NULL when there are no detailed statistics available for the consumer. This can happen for any of the following reasons:

- The consumer is not a wildcarded consumer.
- Detailed statistics are disabled in the server.
- Detailed statistics were not included into this consumerInfo object by the function tibemsAdmin\_GetConsumers that was used to obtain the object.

## See Also

[tibemsAdmin\\_GetConsumers](#)

[tibemsCollection](#)

## tibemsConsumerInfo\_GetDurableName

*Function*

### Purpose

Get the name of the consumer's durable subscription.

### C Declaration

```
tibems_status tibemsConsumerInfo_GetDurableName(
  tibemsConsumerInfo consumerInfo,
  char* name,
  tibems_int name_len);
```

## COBOL Call

```
CALL "tibemsConsumerInfo_GetDurableName"
  USING BY VALUE consumerInfo,
        BY REFERENCE durableName,
        BY VALUE name_len,
        RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
consumerInfo	Get the subscription of this consumer.
name	Store durable name.
name_len	The length of the name buffer.

## Remarks

Return the name of the consumer's durable subscription. Only durable topic consumers have a durable name. The function returns NULL if the consumer is a non-durable topic subscriber or a queue receiver.

## tibemsConsumerInfo\_GetElapsedSinceLastAcknowledged

Function

### Purpose

Get the approximate number of milliseconds elapsed since the last time a message sent to this consumer was acknowledged by the consumer's session.

### C Declaration

```
tibems_status tibemsConsumerInfo_GetElapsedSinceLastAcknowledged(
    tibemsConsumerInfo consumerInfo,
    tibems_long* time);
```

### COBOL Call

```
CALL "tibemsConsumerInfo_GetElapsedSinceLastAcknowledged"
  USING BY VALUE consumerInfo,
        BY REFERENCE time,
        RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
consumerInfo	Get the milliseconds elapsed for this consumer.
time	Stores the elapsed time.

## Remarks

This function gets the approximate number of milliseconds that have elapsed since last time a message sent to this consumer was acknowledged by consumer's session. This value, while returned in milliseconds, has a precision of 1 second. This value should be used for informational purposes only. For example, it can be used to identify consumers which receive messages but do not acknowledge them for some reason.

## tibemsConsumerInfo\_GetElapsedSinceLastSent

*Function*

### Purpose

Get the approximate number of milliseconds elapsed since last time the server sent a message to this consumer.

### C Declaration

```
tibems_status tibemsConsumerInfo_GetElapsedSinceLastSent(
    tibemsConsumerInfo consumerInfo,
    tibems_long* time);
```

### COBOL Call

```
CALL "tibemsConsumerInfo_GetElapsedSinceLastSent"
    USING BY VALUE consumerInfo,
         BY REFERENCE time,
         RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
consumerInfo	Get the milliseconds elapsed for this consumer.
time	The function stores the elapsed milliseconds in this location.

### Remarks

This function gets the approximate number of milliseconds that have elapsed since last time the server sent a message to this consumer. The value returned, while given in milliseconds, has a precision of 1 second. It should be used for informational purposes only.

## tibemsConsumerInfo\_GetID

*Function*

### Purpose

Get the consumer's unique ID.

### C Declaration

```
tibems_status tibemsConsumerInfo_GetID(
    tibemsConsumerInfo consumerInfo,
    tibems_long* cid);
```

### COBOL Call

```
CALL "tibemsConsumerInfo_GetID"
    USING BY VALUE consumerInfo,
         BY REFERENCE id,
         RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
consumerInfo	Get the unique ID of this consumer.
cid	Store the consumer ID in this location.

**tibemsConsumerInfo\_GetPendingMessageCount***Function***Purpose**

Get the number of pending messages for a topic consumer.

**C Declaration**

```
tibems_status tibemsConsumerInfo_GetPendingMessageCount(
    tibemsConsumerInfo consumerInfo,
    tibems_long* count);
```

**COBOL Call**

```
CALL "tibemsConsumerInfo_GetPendingMessageCount"
    USING BY VALUE consumerInfo,
         BY REFERENCE count,
         RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
consumerInfo	Get the number of pending messages for this consumer.
count	Store the number of pending messages.

**Remarks**

This function can be used to retrieve the number of pending messages for a topic consumer only. For queue consumers, the number of pending messages in the corresponding queue must be obtained from the queue. If the consumer is a queue consumer, the function returns 0 (zero).

**See Also**

[tibemsQueueInfo\\_GetPendingMessageCount](#)

**tibemsConsumerInfo\_GetPendingMessageSize***Function***Purpose**

Get the combined size of pending messages for a topic consumer.

## C Declaration

```
tibems_status tibemsConsumerInfo_GetPendingMessageSize(
    tibemsConsumerInfo consumerInfo,
    tibems_long* size);
```

## COBOL Call

```
CALL "tibemsConsumerInfo_GetPendingMessageSize"
    USING BY VALUE consumerInfo,
          BY REFERENCE size,
          RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
consumerInfo	Get the size for this consumer.
size	Store the size of pending messages.

## Remarks

This function can be used to retrieve the combined size of the pending messages for a topic consumer. If the consumer is a queue consumer, the function returns 0 (zero).

## See Also

[tibemsQueueInfo\\_GetPendingMessageSize](#)

## tibemsConsumerInfo\_GetSharedSubscriptionName

*Function*

### Purpose

Get the shared subscription name for a consumer.

## C Declaration

```
tibems_status tibemsConsumerInfo_GetSharedSubscriptionName(
    tibemsConsumerInfo consumerInfo,
    char* sharedName,
    tibems_int name_len);
```

## COBOL Call

```
CALL "tibemsConsumerInfo_GetSharedSubscriptionName"
    USING BY VALUE consumerInfo,
          BY REFERENCE sharedName,
          BY VALUE name-len,
          RETURNING tibems-status
END-CALL.
```



sharedName has usage pointer.

## Parameters

Parameter	Description
consumerInfo	Get the subscription name for this consumer.
sharedName	Store the shared subscription name.
name_len	Length of the sharedName buffer.

## Remarks

This function can be used to retrieve the shared subscription name for the consumer. Only shared consumers have a shared subscription name. The function returns NULL if the consumer is not shared or is a queue receiver.

For more information, see *Shared Subscriptions for Topics* in the *TIBCO Enterprise Message Service User's Guide*.

## See Also

[tibemsConsumerInfo\\_IsShared](#)

## tibemsConsumerInfo\_GetStatistics

*Function*

### Purpose

Get total statistics for a consumer.

### C Declaration

```
tibems_status tibemsConsumerInfo_GetStatistics(
    tibemsConsumerInfo consumerInfo,
    tibemsStatData* statData);
```

### COBOL Call

```
CALL "tibemsConsumerInfo_GetStatistics"
    USING BY VALUE consumerInfo,
          BY REFERENCE stat,
          RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
consumerInfo	Get statistics for this consumer.
statData	Store the statistics.

## Remarks

This function returns NULL when there are no statistics available for the consumer. This can happen for any of the following reasons:

- Statistics are disabled in the server.

- Statistics were not included into this consumerInfo object by the function `tibemsAdmin_GetConsumers` that was used to obtain the object.

### See Also

[tibemsAdmin\\_GetConsumers](#)

[tibemsStatData](#)

## tibemsConsumerInfo\_GetTotalAcknowledgedCount

*Function*

### Purpose

Get the total number of messages which were delivered to this consumer and have been acknowledged by the consumer's session.

### C Declaration

```
tibems_status tibemsConsumerInfo_GetTotalAcknowledgedCount(
    tibemsConsumerInfo consumerInfo,
    tibems_long* count);
```

### COBOL Call

```
CALL "tibemsConsumerInfo_GetTotalAcknowledgedCount"
    USING BY VALUE consumerInfo,
         BY REFERENCE count,
         RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
consumerInfo	Get the number of messages for this consumer.
count	Store the message count in this location.

## tibemsConsumerInfo\_GetTotalMsgCountSentByServer

*Function*

### Purpose

Get the total number of messages the server sent to this consumer since the consumer was created.

### C Declaration

```
tibems_status tibemsConsumerInfo_GetTotalMsgCountSentByServer(
    tibemsConsumerInfo consumerInfo,
    tibems_long* count);
```

### COBOL Call

```
CALL "tibemsConsumerInfo_GetTotalMsgCountSentByServer"
    USING BY VALUE consumerInfo,
         BY REFERENCE count,
         RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
consumerInfo	Get the number of messages for this consumer.
count	Store the message count.

## Remarks

This function returns total number of messages the server sent to this consumer since consumer was created. This value include duplicates of messages that were resent after a consumer's session recovery or rollback. Therefore, the count may not represent true number of unique messages received by this consumer and should be used only for statistical and informational purposes.

## tibemsConsumerInfo\_IsActive

*Function*

### Purpose

Get the active status of the consumer.

### C Declaration

```
tibems_status tibemsConsumerInfo_IsActive(
    tibemsConsumerInfo consumerInfo,
    tibems_bool* active);
```

### COBOL Call

```
CALL "tibemsConsumerInfo_IsActive"
  USING BY VALUE consumerInfo,
        BY REFERENCE active,
        RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
consumerInfo	Get the status of this consumer.
active	Store consumer status.

## Remarks

If the consumer is active, the active status is TRUE. Otherwise, active is FALSE. A consumer is active if the server can send messages to it. Only queue consumers which have never called a receive function remain in inactive state.

Queue consumers which called have called [tibemsMsgConsumer\\_Receive](#), [tibemsMsgConsumer\\_ReceiveNoWait](#), or [tibemsMsgConsumer\\_ReceiveTimeout](#) at least once or are configured with a message callback, and all topic consumers are always active. This function can identify inactive queue consumers which have never called a receive function which and have never received any messages from the server, even when pending messages exist in the corresponding queue.

**See Also**

[tibemsMsgConsumer\\_Receive](#)  
[tibemsMsgConsumer\\_ReceiveNoWait](#)  
[tibemsMsgConsumer\\_ReceiveTimeout](#)

**tibemsConsumerInfo\_IsConnected***Function***Purpose**

Get the connection status of the consumer.

**C Declaration**

```
tibems_status tibemsConsumerInfo_IsConnected(
    tibemsConsumerInfo consumerInfo,
    tibems_bool* connected);
```

**COBOL Call**

```
CALL "tibemsConsumerInfo_IsConnected"
    USING BY VALUE consumerInfo,
         BY REFERENCE connected,
         RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
consumerInfo	Get the status of this consumer.
connected	Store the connection status in this location.

**Remarks**

The connection status will be set to TRUE if this consumer is connected to the server, and FALSE otherwise. Only durable topic subscribers may be in a disconnected state. This function always sets the connection status to TRUE for queue receivers and non-durable topic consumers.

**tibemsConsumerInfo\_IsConnectionConsumer***Function***Purpose**

Get whether this is connection consumer.

**C Declaration**

```
tibems_status tibemsConsumerInfo_IsConnectionConsumer(
    tibemsConsumerInfo consumerInfo,
    tibems_bool* connectionConsumer);
```

**COBOL Call**

```
CALL "tibemsConsumerInfo_IsConnectionConsumer"
    USING BY VALUE consumerInfo,
         BY REFERENCE connectionConsumer,
```

```
RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
consumerInfo	Get whether this consumer is a connection consumer.
connectionConsumer	Store consumer connection status. TRUE indicates that the consumer is a connection consumer.

### Remarks

Set the connection status to TRUE if the consumer is a connection consumer, and FALSE otherwise. Notice that for disconnected durable topic subscribers the function returns FALSE even if the durable was created as connection consumer.

## tibemsConsumerInfo\_IsShared

*Function*

### Purpose

Get the shared subscription status of the consumer.

### C Declaration

```
tibems_status tibemsConsumerInfo_IsShared(
    tibemsConsumerInfo consumerInfo,
    tibems_bool* shared);
```

### COBOL Call

```
CALL "tibemsConsumerInfo_IsShared"
    USING BY VALUE consumerInfo,
         BY REFERENCE shared,
         RETURNING tibems-status
END-CALL.
```



shared has usage pointer.

### Parameters

Parameter	Description
consumerInfo	Get the shared status of this consumer.
shared	Store the consumer's shared status in this location.

### Remarks

If the consumer is shared, the shared status will be set to TRUE. Otherwise, shared is FALSE.

For more information, see *Shared Subscriptions for Topics* in the *TIBCO Enterprise Message Service User's Guide*.

### See Also

[tibemsConsumerInfo\\_GetSharedSubscriptionName](#)

## tibemsDetailedDestStat

*Type*

### Purpose

Represent detailed destination statistics about objects such as topics, message consumers, or message producers.

### Remarks

Detailed statistics are optionally collected by the server for wildcarded consumers, unidentified producers and routes.

Function	Description
<a href="#">tibemsDetailedDestStat_GetDestinationName</a>	Get the name of the destination.
<a href="#">tibemsDetailedDestStat_GetDestinationType</a>	Get the type of the destination.
<a href="#">tibemsDetailedDestStat_GetDestinationType</a>	Get statistics for a unidirectional destination.

### Related Types

[tibemsStatData](#)

### See Also

[tibemsMsg\\_GetDestination](#)

## tibemsDetailedDestStat\_GetDestinationName

*Function*

### Purpose

Get the name of the destination.

### C Declaration

```
tibems_status tibemsDetailedDestStat_GetDestinationName(
    tibemsDetailedDestStat detailedDestStat,
    char* name,
    tibems_int name_len);
```

### COBOL Call

```
CALL "tibemsDetailedDestStat_GetDestinationName"
  USING BY VALUE detailedDestStat,
        BY REFERENCE destName,
        BY VALUE name_len,
        RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
detailedDestStat	Get the name of this destination.
name	Store destination name.
name_len	Length of the name buffer.

## See Also

[tibemsDestinationType](#)

## tibemsDetailedDestStat\_GetDestinationType

*Function*

### Purpose

Get the type of the destination.

### C Declaration

```
tibems_status tibemsDetailedDestStat_GetDestinationType(
    tibemsDetailedDestStat detailedDestStat,
    tibemsDestinationType* destType);
```

### COBOL Call

```
CALL "tibemsDetailedDestStat_GetDestinationType"
    USING BY VALUE detailedDestStat,
         BY REFERENCE destType,
         RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
detailedDestStat	Get the type of this destination.
destType	Store destination type.

## See Also

[tibemsDestinationType](#)

## tibemsDetailedDestStat\_GetStatData

*Function*

### Purpose

Get statistics for a unidirectional destination.

## C Declaration

```
tibems_status tibemsDetailedDestStat_GetStatData(
    tibemsDetailedDestStat detailDestStat,
    tibemsStatData* statData);
```

## COBOL Call

```
CALL "tibemsDetailedDestStat_GetStatData"
    USING BY VALUE detailedDestStat,
         BY REFERENCE stat,
         RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
detailedDestStat	Get statistics for this destination.
statData	The function stores the statistics in this location.

## Remarks

This function retrieves unidirectional destination statistics. Unidirectional destinations statistics are collected for both producers and consumers.

## tibemsProducerInfo

*Type*

### Purpose

Represent a message producer.

Function	Description
<a href="#">tibemsProducerInfo_Destroy</a>	Destroy a producerInfo object.
<a href="#">tibemsProducerInfo_GetCreateTime</a>	Get the producer create time in milliseconds.
<a href="#">tibemsProducerInfo_GetDestinationName</a>	Get the producer's destination name.
<a href="#">tibemsProducerInfo_GetDestinationType</a>	Get the producer's destination type.
<a href="#">tibemsProducerInfo_GetDetailedStatistics</a>	Get detailed statistics for an unidentified producer.
<a href="#">tibemsProducerInfo_GetID</a>	Get the producer's ID.
<a href="#">tibemsProducerInfo_GetStatistics</a>	Get total statistics for a producer.

## Related Types

[tibemsDetailedDestStat](#)

[tibemsStatData](#)

## tibemsProducerInfo\_Destroy

*Function*

### Purpose

Destroy a producerInfo object.

### C Declaration

```
tibems_status tibemsProducerInfo_Destroy(
    tibemsProducerInfo producerInfo);
```

### COBOL Call

```
CALL "tibemsProducerInfo_Destroy"
    USING BY VALUE producerInfo,
         RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
producerInfo	The <a href="#">tibemsProducerInfo</a> object to destroy.

## tibemsProducerInfo\_GetCreateTime

*Function*

### Purpose

Get the producer create time in milliseconds.

### C Declaration

```
tibems_status tibemsProducerInfo_GetCreateTime(
    tibemsProducerInfo producerInfo,
    tibems_long* created);
```

### COBOL Call

```
CALL "tibemsProducerInfo_GetCreateTime"
    USING BY VALUE producerInfo,
         BY REFERENCE created,
         RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
producerInfo	Get the create time for this producer.
created	Store the create time.

## tibemsProducerInfo\_GetDestinationName

*Function*

### Purpose

Get the producer's destination name.

### C Declaration

```
tibems_status tibemsProducerInfo_GetDestinationName(
    tibemsProducerInfo producerInfo,
    char* name
    tibems_int name_len);
```

### COBOL Call

```
CALL "tibemsProducerInfo_GetDestinationName"
  USING BY VALUE producerInfo,
        BY REFERENCE destName,
        BY VALUE name_len,
        RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
producerInfo	Get the destination name of this producer.
name	Store destination name.
name_len	Length of the name buffer.

## tibemsProducerInfo\_GetDestinationType

*Function*

### Purpose

Get the producer's destination type.

### C Declaration

```
tibems_status tibemsProducerInfo_GetDestinationType(
    tibemsProducerInfo producerInfo,
    tibemsDestinationType* type);
```

### COBOL Call

```
CALL "tibemsProducerInfo_GetDestinationType"
  USING BY VALUE producerInfo,
        BY REFERENCE destType,
        RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
producerInfo	Get the destination type of this producer.

Parameter	Description
type	Store destination type.

**See Also**

[tibemsDestinationType](#)

**tibemsProducerInfo\_GetDetailedStatistics**

*Function*

**Purpose**

Get detailed statistics for an unidentified producer.

**C Declaration**

```
tibems_status tibemsProducerInfo_GetDetailedStatistics(
    tibemsProducerInfo producerInfo,
    tibemsCollection* collection;
```

**COBOL Call**

```
CALL "tibemsProducerInfo_GetDetailedStatistics"
    USING BY VALUE producerInfo,
         BY REFERENCE collection,
         RETURNING tibems-status
END-CALL.
```



collection has usage pointer.

**Parameters**

Parameter	Description
producerInfo	Get statistics for this producer.
collection	Store the statistics as a collection of <a href="#">tibemsDetailedDestStat</a> objects.

**Remarks**

Get detailed statistics for an unidentified producer, giving a breakdown of the producer's aggregate statistics across all destinations that it has published messages on.

This function returns `NULL` when there are no detailed statistics available for the producer. This can happen when, for example, detailed statistics for producers are disabled in the server.

**See Also**

[tibemsDetailedDestStat](#)

**tibemsProducerInfo\_GetID**

*Function*

**Purpose**

Get the producer's ID.

**C Declaration**

```
tibems_status tibemsProducerInfo_GetID(
    tibemsProducerInfo producerInfo,
    tibems_long* id);
```

**COBOL Call**

```
CALL "tibemsProducerInfo_GetID"
    USING BY VALUE producerInfo,
         BY REFERENCE id,
         RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
producerInfo	Get the ID for this producer.
id	Stores the producer ID.

**tibemsProducerInfo\_GetStatistics**

*Function*

**Purpose**

Get total statistics for a producer.

**C Declaration**

```
tibems_status tibemsProducerInfo_GetStatistics(
    tibemsProducerInfo producerInfo,
    tibemsStatData* statData);
```

**COBOL Call**

```
CALL "tibemsProducerInfo_GetStatistics"
    USING BY VALUE producerInfo,
         BY REFERENCE stat,
         RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
producerInfo	Get statistics for this producer.
statData	Store the statistics. The stored value will be NULL if statistics are disabled in the server and so there are none available for the producer.

**See Also**

[tibemsStatData](#)

## tibemsQueueInfo

*Type*

### Purpose

Represent a message queue that is configured in the EMS server.

Function	Description
<code>tibemsQueueInfo_Create</code>	Create a <code>tibemsQueueInfo</code> object.
<code>tibemsQueueInfo_Destroy</code>	Destroy a <code>tibemsQueueInfo</code> object.
<code>tibemsQueueInfo_GetDeliveredMessageCount</code>	Get the total number of messages that have been delivered to consumer applications but have not yet been acknowledged.
<code>tibemsQueueInfo_GetFlowControlMaxBytes</code>	Get the volume of pending messages at which flow control is enabled for the queue.
<code>tibemsQueueInfo_GetInboundStatistics</code>	Get inbound statistics for this queue.
<code>tibemsQueueInfo_GetMaxBytes</code>	Get the maximum number of bytes that the server stores of pending messages bound for this queue.
<code>tibemsQueueInfo_GetMaxMsgs</code>	Get the maximum number of pending messages bound for the queue that the server will store.
<code>tibemsQueueInfo_GetName</code>	Get the name of this queue.
<code>tibemsQueueInfo_GetOutboundStatistics</code>	Get outbound statistics for this queue.
<code>tibemsQueueInfo_GetOverflowPolicy</code>	Get the overflow policy for this queue.
<code>tibemsQueueInfo_GetPendingMessageCount</code>	Get the total number of pending messages for this queue.
<code>tibemsQueueInfo_GetPendingMessageSize</code>	Get the total size of all pending messages for this queue.
<code>tibemsQueueInfo_GetPendingPersistentMessageCount</code>	Get the total number of pending messages for this queue that were sent persistently.
<code>tibemsQueueInfo_GetPendingPersistentMessageSize</code>	Get the total size of all pending messages for this queue that were sent persistently.

Function	Description
<a href="#">tibemsQueueInfo_GetReceiverCount</a>	Get the number of active receivers on this queue.

### Related Types

[tibemsQueue](#)

[tibemsStatData](#)

## tibemsQueueInfo\_Create

*Function*

### Purpose

Create a `tibemsQueueInfo` object.

### C Declaration

```
tibems_status tibemsQueueInfo_Create(
    tibemsQueueInfo* queueInfo,
    const char* queueName);
```

### COBOL Call

```
CALL "tibemsQueueInfo_Create"
  USING BY REFERENCE queueInfo,
        BY REFERENCE name,
        RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
<code>queueInfo</code>	Store the new <code>tibemsQueueInfo</code> object in this location.
<code>queueName</code>	Create the <code>tibemsQueueInfo</code> object with this name. The <code>queueName</code> can include wildcards.

### Remarks

This function is used to create a `tibemsQueueInfo` object with the specified name. The `tibemsQueueInfo` object may then be passed as the `tibemsDestinationInfo` argument to the [tibemsAdmin\\_GetConsumers](#) and [tibemsAdmin\\_GetProducerStatistics](#) functions. The name may include wildcards.

For more information on wildcards, see *Wildcards* in the *TIBCO Enterprise Message Service User's Guide*.

### See Also

[tibemsTopicInfo\\_Create](#)

[tibemsAdmin\\_GetConsumers](#)

[tibemsAdmin\\_GetProducerStatistics](#)

## tibemsQueueInfo\_Destroy

*Function*

### Purpose

Destroy a `tibemsQueueInfo` object.

### C Declaration

```
tibems_status tibemsQueueInfo_Destroy(
    tibemsQueueInfo queueInfo);
```

### COBOL Call

```
CALL "tibemsQueueInfo_Destroy"
    USING BY VALUE queueInfo,
         RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
<code>queueInfo</code>	The <code>tibemsQueueInfo</code> object to destroy.

### See Also

[tibemsTopicInfo\\_Destroy](#)

## tibemsQueueInfo\_GetDeliveredMessageCount

*Function*

### Purpose

Get the total number of messages that have been delivered to consumer applications but have not yet been acknowledged.

### C Declaration

```
tibems_status tibemsQueueInfo_GetDeliveredMessageCount(
    tibemsQueueInfo queueInfo,
    tibems_long* count);
```

### COBOL Call

```
CALL "tibemsQueueInfo_GetDeliveredMessageCount"
    USING BY VALUE queueInfo,
         BY REFERENCE count,
         RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
<code>queueInfo</code>	Get the number of unacknowledged messages in this queue.
<code>count</code>	Store the message count.

## tibemsQueueInfo\_GetFlowControlMaxBytes

*Function*

### Purpose

Get the volume of pending messages at which flow control is enabled for the queue.

### C Declaration

```
tibems_status tibemsQueueInfo_GetFlowControlMaxBytes(
    tibemsQueueInfo queueInfo,
    tibems_long* maxBytes);
```

### COBOL Call

```
CALL "tibemsQueueInfo_GetFlowControlMaxBytes"
  USING BY VALUE queueInfo,
        BY REFERENCE maxBytes,
        RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
queueInfo	Get the volume for this queue.
maxBytes	Store the volume.  The value stored indicates the volume of pending messages, in bytes, that the server will store in the queue before enabling flow control. A value of 0 indicates that flow control will never be enabled.

### See Also

[tibemsTopicInfo\\_GetFlowControlMaxBytes](#)

## tibemsQueueInfo\_GetInboundStatistics

*Function*

### Purpose

Get inbound statistics for this queue.

### C Declaration

```
tibems_status tibemsQueueInfo_GetInboundStatistics(
    tibemsQueueInfo queueInfo,
    tibemsStatData* statData);
```

### COBOL Call

```
CALL "tibemsQueueInfo_GetInboundStatistics"
  USING BY VALUE queueInfo,
        BY REFERENCE statData,
        RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
queueInfo	Get statistics for this queue.
statData	Store the statistics.

**Remarks**

This function retrieves the inbound statistics for the queue. Inbound statistics include all messages sent into the queue by EMS clients and routed servers.

**See Also**

[tibemsTopicInfo\\_GetInboundStatistics](#)

**tibemsQueueInfo\_GetMaxBytes**

*Function*

**Purpose**

Get the maximum number of bytes that the server stores of pending messages bound for this queue.

**C Declaration**

```
tibems_status tibemsQueueInfo_GetMaxBytes(
    tibemsQueueInfo queueInfo,
    tibems_long* maxBytes);
```

**COBOL Call**

```
CALL "tibemsQueueInfo_GetMaxBytes"
  USING BY VALUE queueInfo,
        BY REFERENCE maxBytes,
        RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
queueInfo	Get the maximum bytes for this queue.
maxBytes	Store the number of bytes.

**See Also**

[tibemsTopicInfo\\_GetMaxBytes](#)

**tibemsQueueInfo\_GetMaxMsgs**

*Function*

**Purpose**

Get the maximum number of pending messages bound for the queue that the server will store.

## C Declaration

```
tibems_status tibemsQueueInfo_GetMaxMsgs(
    tibemsQueueInfo queueInfo,
    tibems_long* maxMsgs);
```

## COBOL Call

```
CALL "tibemsQueueInfo_GetMaxMsgs"
    USING BY VALUE queueInfo,
         BY REFERENCE maxMsgs,
         RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
queueInfo	Maximum number of pending messages.
maxMsgs	Store the maximum number of messages.

## See Also

[tibemsTopicInfo\\_GetMaxMsgs](#)

## tibemsQueueInfo\_GetName

*Function*

### Purpose

Get the name of this queue.

## C Declaration

```
tibems_status tibemsQueueInfo_GetName(
    tibemsQueueInfo queueInfo,
    char* name,
    tibems_int name_len);
```

## COBOL Call

```
CALL "tibemsQueueInfo_GetName"
    USING BY VALUE queueInfo,
         BY REFERENCE name,
         BY VALUE name_len,
         RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
queueInfo	Get the name of this queue.
name	Storesthe queue name.
name_len	Length of the name buffer.

**See Also**

[tibemsTopicInfo\\_GetName](#)

**tibemsQueueInfo\_GetOutboundStatistics**

*Function*

**Purpose**

Get outbound statistics for this queue.

**C Declaration**

```
tibems_status tibemsQueueInfo_GetOutboundStatistics(
    tibemsQueueInfo queueInfo,
    tibemsStatData* statData);
```

**COBOL Call**

```
CALL "tibemsQueueInfo_GetOutboundStatistics"
    USING BY VALUE queueInfo,
         BY REFERENCE statData,
         RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
queueInfo	Get statistics for this queue.
statData	Store the statistics.

**Remarks**

This function retrieves the outbound statistics for the queue. Outbound statistics include all messages delivered from the queue to EMS clients and routed servers.

**See Also**

[tibemsTopicInfo\\_GetOutboundStatistics](#)

**tibemsQueueInfo\_GetOverflowPolicy**

*Function*

**Purpose**

Get the overflow policy for this queue.

**C Declaration**

```
tibems_status tibemsQueueInfo_GetOverflowPolicy(
    tibemsQueueInfo queueInfo,
    tibems_int* overflowPolicy);
```

**COBOL Call**

```
CALL "tibemsQueueInfo_GetOverflowPolicy"
    USING BY VALUE queueInfo,
         BY REFERENCE overflowPolicy,
```

```
RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
queueInfo	Get the policy for this queue.
overflowPolicy	Store the overflow policy.

### Remarks

This function retrieves the overflow policy for the queue. Possible values are:

- TIBEMS\_OVERFLOW\_DEFAULT
- TIBEMS\_OVERFLOW\_DISCARD\_OLD
- TIBEMS\_OVERFLOW\_REJECT\_INCOMING

For more information about overflow policies, see the *TIBCO Enterprise Message Service User's Guide*.

### See Also

[tibemsTopicInfo\\_GetOverflowPolicy](#)

## tibemsQueueInfo\_GetPendingMessageCount

*Function*

### Purpose

Get the total number of pending messages for this queue.

### C Declaration

```
tibems_status tibemsQueueInfo_GetPendingMessageCount(
    tibemsQueueInfo queueInfo,
    tibems_long* count);
```

### COBOL Call

```
CALL "tibemsQueueInfo_GetPendingMessageCount"
    USING BY VALUE queueInfo,
         BY REFERENCE count,
         RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
queueInfo	Get the number of messages for this queue.
count	Store the number of messages.

### See Also

[tibemsTopicInfo\\_GetPendingMessageCount](#)

## tibemsQueueInfo\_GetPendingMessageSize

*Function*

### Purpose

Get the total size of all pending messages for this queue.

### C Declaration

```
tibems_status tibemsQueueInfo_GetPendingMessageSize(
    tibemsQueueInfo queueInfo,
    tibems_long* size);
```

### COBOL Call

```
CALL "tibemsQueueInfo_GetPendingMessageSize"
    USING BY VALUE queueInfo,
         BY REFERENCE size,
         RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
queueInfo	Get the size of pending messages for this queue.
size	Store the size of pending messages.

### See Also

[tibemsTopicInfo\\_GetPendingMessageSize](#)

## tibemsQueueInfo\_GetPendingPersistentMessageCount

*Function*

### Purpose

Get the total number of pending messages for this queue that were sent persistently.

### C Declaration

```
tibems_status tibemsQueueInfo_GetPendingPersistentMessageCount(
    tibemsQueueInfo queueInfo,
    tibems_long* count);
```

### COBOL Call

```
CALL "tibemsQueueInfo_GetPendingPersistentMessageCount"
    USING BY VALUE queueInfo,
         BY REFERENCE count,
         RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
queueInfo	Get the number of messages for this queue.

Parameter	Description
count	Store the number of messages.

**See Also**

[tibemsTopicInfo\\_GetPendingPersistentMessageCount](#)

**tibemsQueueInfo\_GetPendingPersistentMessageSize**

*Function*

**Purpose**

Get the total size of all pending messages for this queue that were sent persistently.

**C Declaration**

```
tibems_status tibemsQueueInfo_GetPendingPersistentMessageSize(
    tibemsQueueInfo queueInfo,
    tibems_long* size);
```

**COBOL Call**

```
CALL "tibemsQueueInfo_GetPendingPersistentMessageSize"
  USING BY VALUE queueInfo,
        BY REFERENCE size,
        RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
queueInfo	Get the size of pending messages for this queue.
size	Store the size of pending messages.

**See Also**

[tibemsTopicInfo\\_GetPendingPersistentMessageSize](#)

**tibemsQueueInfo\_GetReceiverCount**

*Function*

**Purpose**

Get the number of active receivers on this queue.

**C Declaration**

```
tibems_status tibemsQueueInfo_GetReceiverCount(
    tibemsQueueInfo queueInfo,
    tibems_int* count);
```

**COBOL Call**

```
CALL "tibemsQueueInfo_GetReceiverCount"
  USING BY VALUE queueInfo,
        BY REFERENCE count,
```

```
RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
queueInfo	Get the number of receivers on this queue.
count	Store the number of receivers.

## tibemsServerInfo

*Type*

### Purpose

Represent a TIBCO Enterprise Message Service server.

### Remarks

This type represents metrics for an EMS server.

Function	Description
<a href="#">tibemsServerInfo_Destroy</a>	Destroy a <code>tibemsServerInfo</code> object.
<a href="#">tibemsServerInfo_GetConsumerCount</a>	Get the total number of consumers.
<a href="#">tibemsServerInfo_GetProducerCount</a>	Get the total number of producers.
<a href="#">tibemsServerInfo_GetQueueCount</a>	Get the total number of queues in the server.
<a href="#">tibemsServerInfo_GetTopicCount</a>	Get the total number of topics in the server.

## tibemsServerInfo\_Destroy

*Function*

### Purpose

Destroy a `tibemsServerInfo` object.

### C Declaration

```
tibems_status tibemsServerInfo_Destroy(
    tibemsServerInfo serverInfo);
```

### COBOL Call

```
CALL "tibemsServerInfo_Destroy"
    USING BY VALUE serverInfo,
    RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
serverInfo	The tibemsServerInfo object to destroy.

**tibemsServerInfo\_GetConsumerCount***Function***Purpose**

Get the total number of consumers.

**C Declaration**

```
tibems_status tibemsServerInfo_GetConsumerCount(
    tibemsServerInfo serverInfo,
    tibems_int* count);
```

**COBOL Call**

```
CALL "tibemsServerInfo_GetConsumerCount"
    USING BY VALUE serverInfo,
         BY REFERENCE count,
         RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
serverInfo	Get the number of consumers for this server.
count	Store the number of consumers.

**tibemsServerInfo\_GetProducerCount***Function***Purpose**

Get the total number of producers.

**C Declaration**

```
tibems_status tibemsServerInfo_GetProducerCount(
    tibemsServerInfo serverInfo,
    tibems_int* count);
```

**COBOL Call**

```
CALL "tibemsServerInfo_GetProducerCount"
    USING BY VALUE serverInfo,
         BY REFERENCE count,
         RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
serverInfo	Get the number of producers for this server.
count	Store the number of producers.

**tibemsServerInfo\_GetQueueCount***Function***Purpose**

Get the total number of queues in the server.

**C Declaration**

```
tibems_status tibemsServerInfo_GetQueueCount(
    tibemsServerInfo serverInfo,
    tibems_int* count);
```

**COBOL Call**

```
CALL "tibemsServerInfo_GetQueueCount"
    USING BY VALUE serverInfo,
         BY REFERENCE count,
         RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
serverInfo	Get the number of queues in this server.
count	Store the number of queues.

**tibemsServerInfo\_GetTopicCount***Function***Purpose**

Get the total number of topics in the server.

**C Declaration**

```
tibems_status tibemsServerInfo_GetTopicCount(
    tibemsServerInfo serverInfo,
    tibems_int* count);
```

**COBOL Call**

```
CALL "tibemsServerInfo_GetTopicCount"
    USING BY VALUE serverInfo,
         BY REFERENCE count,
         RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
serverInfo	Get the number of topics in this server.
count	Store the number of topics.

**tibemsStatData***Type***Purpose**

Represent statistical data about another object, such as a topic or queue.

**Remarks**

Statistical data contains the total number of messages and their cumulative size. It can also provide message rates, as the number of messages sent each second and number of bytes each second. Whether or not rate information is available is controlled by the server configuration. If rate collection is turned off, all rate numbers are set to 0. However, in those cases the application can calculate the rates based on absolute numbers taken at periodic intervals.

Function	Description
<a href="#">tibemsStatData_GetByteRate</a>	Get the average rate of bytes sent or received each second.
<a href="#">tibemsStatData_GetMessageRate</a>	Get the rate of messages sent or received each second.
<a href="#">tibemsStatData_GetTotalBytes</a>	Get the total size of messages sent or received each second.
<a href="#">tibemsStatData_GetTotalMessages</a>	Get the total number of messages sent or received each second.

**Related Types**[tibemsQueue](#)[tibemsTopic](#)[tibemsConsumerInfo](#)[tibemsProducerInfo](#)[tibemsQueueInfo](#)[tibemsTopicInfo](#)**tibemsStatData\_GetByteRate***Function***Purpose**

Get the average rate of bytes sent or received each second.

## C Declaration

```
tibems_status tibemsStatData_GetByteRate(
    tibemsStatData statData,
    tibems_long* byteRate);
```

## COBOL Call

```
CALL "tibemsStatData_GetByteRate"
    USING BY VALUE statData,
         BY REFERENCE rate_size,
         RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
statData	Get the rate for this message consumer, producer, topic, or queue.
byteRate	Store the average byte rate.

## Remarks

This function gets the average byte rate for messages sent or received per second. For message consumers, this rate reflects the number of messages received by the consumer. For messages producers, it is the rate of messages sent, and for topics and queues the rate includes both inbound and outbound messages.

## tibemsStatData\_GetMessageRate

*Function*

### Purpose

Get the rate of messages sent or received each second.

## C Declaration

```
tibems_status tibemsStatData_GetMessageRate(
    tibemsStatData statData,
    tibems_long* messageRate);
```

## COBOL Call

```
CALL "tibemsStatData_GetMessageRate"
    USING BY VALUE statData,
         BY REFERENCE rate_msgs,
         RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
statData	Get the rate for this message consumer, producer, topic, or queue.
messageRate	Store the average message rate.

**Remarks**

This function gets the average number of messages sent or received per second. For message consumers, this rate reflects the number of messages received by the consumer. For messages producers, it is the rate of messages sent, and for topics and queues the rate includes both inbound and outbound messages.

**tibemsStatData\_GetTotalBytes**

*Function*

**Purpose**

Get the total size of messages sent or received each second.

**C Declaration**

```
tibems_status tibemsStatData_GetTotalBytes(
    tibemsStatData statData,
    tibems_long* bytes);
```

**COBOL Call**

```
CALL "tibemsStatData_GetTotalBytes"
    USING BY VALUE statData,
         BY REFERENCE size,
         RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
statData	Get the total message size for this consumer, producer, topic, or queue.
bytes	Store the size of messages.

**Remarks**

This function gets the total size of messages sent or received. For message consumers, this is the size of messages received by the consumer. For messages producers, it is the size of messages sent, and for topics and queues the size includes both inbound and outbound messages.

**tibemsStatData\_GetTotalMessages**

*Function*

**Purpose**

Get the total number of messages sent or received each second.

**C Declaration**

```
tibems_status tibemsStatData_GetTotalMessages(
    tibemsStatData statData,
    tibems_long* messages);
```

**COBOL Call**

```
CALL "tibemsStatData_GetTotalMessages"
    USING BY VALUE statData,
         BY REFERENCE msgs,
```

RETURNING tibems-status  
END-CALL.

### Parameters

Parameter	Description
statData	Get the total number of messages for this consumer, producer, topic, or queue.
messages	Store the number of messages.

### Remarks

This function gets the total number of messages sent or received. For message consumers, this is the number of messages received by the consumer. For messages producers, it is the number of messages sent, and for topics and queues the number includes both inbound and outbound messages.

## tibemsSubscriptionInfo

*Type*

### Purpose

Represent a subscription in the server.

Function	Description
<code>tibemsSubscriptionInfo_Destroy</code>	Destroy a subscriptionInfo object.
<code>tibemsSubscriptionInfo_GetConsumerCount</code>	Get the number of consumers for this subscription.
<code>tibemsSubscriptionInfo_GetCreateTime</code>	Get a subscription's creation time in milliseconds.
<code>tibemsSubscriptionInfo_GetID</code>	Get the subscription's unique ID.
<code>tibemsSubscriptionInfo_GetName</code>	Get the name of the subscription.
<code>tibemsSubscriptionInfo_GetPendingMessageCount</code>	Get the number of pending messages for a subscription.
<code>tibemsSubscriptionInfo_GetPendingMessageSize</code>	Get the combined size of pending messages for a subscription.
<code>tibemsSubscriptionInfo_GetSelector</code>	Return this subscription's selector.
<code>tibemsSubscriptionInfo_GetTopicName</code>	Get the topic name that this subscription is for.
<code>tibemsSubscriptionInfo_HasSelector</code>	Get whether the subscription has a selector.

Function	Description
<a href="#">tibemsSubscriptionInfo_IsDurable</a>	Get whether this is a durable subscription.
<a href="#">tibemsSubscriptionInfo_IsShared</a>	Get whether this is a shared subscription.

### Related Types

[tibemsConsumerInfo](#)

## tibemsSubscriptionInfo\_Destroy

*Function*

### Purpose

Destroy a subscriptionInfo object.

### C Declaration

```
tibems_status tibemsSubscriptionInfo_Destroy(
    tibemsSubscriptionInfo subscriptionInfo)
```

### COBOL Call

```
CALL "tibemsSubscriptionInfo_Destroy"
    USING BY VALUE subscriptionInfo,
           RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
subscriptionInfo	The subscriptionInfo object to be destroyed.

## tibemsSubscriptionInfo\_GetConsumerCount

*Function*

### Purpose

Get the number of consumers for this subscription.

### C Declaration

```
tibems_status tibemsSubscriptionInfo_GetConsumerCount(
    tibemsSubscriptionInfo subscriptionInfo,
    tibems_int* consumerCount);
```

### COBOL Call

```
CALL "tibemsSubscriptionInfo_GetConsumerCount"
    USING BY VALUE subscriptionInfo,
           BY REFERENCE consumerCount,
           RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
subscriptionInfo	Get the number of consumers for this subscription.
consumerCount	Store the number of consumers.

**Remarks**

Returns the number of consumers for this subscription. If the subscription is not shared, the count cannot exceed 1. If the subscription is shared, the count can exceed 1, since a shared subscription can have many shared consumers.

For durable subscriptions (shared and unshared), this count can be 0 if there is no active durable consumer.

**See Also**

[tibemsSubscriptionInfo\\_IsDurable](#)

[tibemsSubscriptionInfo\\_IsShared](#)

**tibemsSubscriptionInfo\_GetCreateTime**

*Function*

**Purpose**

Get a subscription's creation time in milliseconds.

**C Declaration**

```
tibems_status tibemsSubscriptionInfo_GetCreateTime(
    tibemsSubscriptionInfo subscriptionInfo,
    tibems_long* created);
```

**COBOL Call**

```
CALL "tibemsSubscriptionInfo_GetCreateTime"
    USING BY VALUE subscriptionInfo,
         BY REFERENCE created,
         RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
subscriptionInfo	Get the creation time of this subscription.
created	Store the create time in this location.

**tibemsSubscriptionInfo\_GetID**

*Function*

**Purpose**

Get the subscription's unique ID.

**C Declaration**

```
tibems_status tibemsSubscriptionInfo_GetID(
    tibemsSubscriptionInfo subscriptionInfo,
    tibems_long* id);
```

**COBOL Call**

```
CALL "tibemsSubscriptionInfo_GetID"
    USING BY VALUE subscriptionInfo,
         BY REFERENCE id,
         RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
subscriptionInfo	Get the unique ID of this subscription.
id	Store the subscription ID.

**tibemsSubscriptionInfo\_GetName**

*Function*

**Purpose**

Get the name of the subscription.

**C Declaration**

```
tibems_status tibemsSubscriptionInfo_GetName(
    tibemsSubscriptionInfo subscriptionInfo,
    char* name,
    tibems_int* nameLength);
```

**COBOL Call**

```
CALL "tibemsSubscriptionInfo_GetName"
    USING BY VALUE subscriptionInfo,
         BY REFERENCE name,
         BY REFERENCE nameLength,
         RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
subscriptionInfo	Get the name of this subscription.
name	Copy the subscription name in the provided buffer.
nameLength	Length of the name buffer.

## tibemsSubscriptionInfo\_GetPendingMessageCount

*Function*

### Purpose

Get the number of pending messages for a subscription.

### C Declaration

```
tibems_status tibemsSubscriptionInfo_GetPendingMessageCount(
    tibemsSubscriptionInfo subscriptionInfo,
    tibems_long* msgCount);
```

### COBOL Call

```
CALL "tibemsSubscriptionInfo_GetPendingMessageCount"
    USING BY VALUE subscriptionInfo,
         BY REFERENCE msgCount,
         RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
subscriptionInfo	Get the number of pending messages for this subscription.
msgCount	Store the number of pending messages.

### See Also

[tibemsSubscriptionInfo\\_GetPendingMessageSize](#)

## tibemsSubscriptionInfo\_GetPendingMessageSize

*Function*

### Purpose

Get the combined size of pending messages for a subscription.

### C Declaration

```
tibems_status tibemsSubscriptionInfo_GetPendingMessageSize(
    tibemsSubscriptionInfo subscriptionInfo,
    tibems_long* msgSize);
```

### COBOL Call

```
CALL "tibemsSubscriptionInfo_GetPendingMessageSize"
    USING BY VALUE subscriptionInfo,
         BY REFERENCE msgSize,
         RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
subscriptionInfo	Get the combined pending messages size for this subscription.

Parameter	Description
msgSize	Store the combined size of pending messages.

**See Also**

[tibemsSubscriptionInfo\\_GetPendingMessageCount](#)

**tibemsSubscriptionInfo\_GetSelector**

*Function*

**Purpose**

Return this subscription's selector.

**C Declaration**

```
tibems_status tibemsSubscriptionInfo_GetSelector(
    tibemsSubscriptionInfo subscriptionInfo,
    char* selector,
    tibems_int selectorLength);
```

**COBOL Call**

```
CALL "tibemsSubscriptionInfo_GetSelector"
  USING BY VALUE subscriptionInfo,
        BY REFERENCE selector,
        BY VALUE selectorLength,
        RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
subscriptionInfo	Get the selector for this subscription.
selector	Copy the selector into the given buffer. If there is no selector, the empty string is copied.
selectorLength	Length of the selector buffer.

**tibemsSubscriptionInfo\_GetTopicName**

*Function*

**Purpose**

Get the topic name that this subscription is for.

**C Declaration**

```
tibems_status tibemsSubscriptionInfo_GetTopicName(
    tibemsSubscriptionInfo subscriptionInfo,
    char* topicName,
    tibems_int topicNameLength);
```

## COBOL Call

```
CALL "tibemsSubscriptionInfo_GetTopicName"
  USING BY VALUE subscriptionInfo,
        BY REFERENCE topicName,
        BY VALUE topicNameLength,
        RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
subscriptionInfo	Get the topic name that this subscription is for.
topicName	Copy the topic name in this buffer.
topicNameLength	Length of the topicName buffer.

## tibemsSubscriptionInfo\_HasSelector

*Function*

### Purpose

Get whether the subscription has a selector.

### C Declaration

```
tibems_status tibemsSubscriptionInfo_HasSelector(
    tibemsSubscriptionInfo subscriptionInfo,
    tibems_bool* hasSelector);
```

## COBOL Call

```
CALL "tibemsSubscriptionInfo_HasSelector"
  USING BY VALUE subscriptionInfo,
        BY REFERENCE hasSelector,
        RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
subscriptionInfo	Get the selector status for this subscription.
hasSelector	Store the status of the selector. The value is TIBEMS_TRUE if the subscription has a selector, TIBEMS_FALSE if not.

## See Also

[tibemsSubscriptionInfo\\_GetSelector](#)

## tibemsSubscriptionInfo\_IsDurable

*Function*

### Purpose

Get whether this is a durable subscription.

### C Declaration

```
tibems_status tibemsSubscriptionInfo_IsDurable(
    tibemsSubscriptionInfo subscriptionInfo,
    tibems_bool* durable);
```

### COBOL Call

```
CALL "tibemsSubscriptionInfo_IsDurable"
  USING BY VALUE subscriptionInfo,
        BY REFERENCE durable,
        RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
subscriptionInfo	Get whether this subscription is a durable subscription.
durable	Store subscription durable status in this location. TIBEMS_TRUE indicates that the subscription is a durable subscription.

## tibemsSubscriptionInfo\_IsShared

*Function*

### Purpose

Get whether this is a shared subscription.

### C Declaration

```
tibems_status tibemsSubscriptionInfo_IsShared(
    tibemsSubscriptionInfo subscriptionInfo,
    tibems_bool* shared);
```

### COBOL Call

```
CALL "tibemsSubscriptionInfo_IsShared"
  USING BY VALUE subscriptionInfo,
        BY REFERENCE shared,
        RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
subscriptionInfo	Get whether this subscription is a shared subscription.

Parameter	Description
shared	Store subscription shared status. TIBEMS_TRUE indicates that subscription is a shared subscription.

## tibemsTopicInfo

*Type*

### Purpose

Represent a topic that is configured in the server.

Function	Description
<code>tibemsTopicInfo_Create</code>	Create a <code>tibemsTopicInfo</code> object.
<code>tibemsTopicInfo_Destroy</code>	Destroy a <code>tibemsTopicInfo</code> object.
<code>tibemsTopicInfo_GetActiveDurableCount</code>	Get the current number of active durable subscribers for this topic.
<code>tibemsTopicInfo_GetDurableSubscriptionCount</code>	Get the current number of durable subscriptions for this topic.
<code>tibemsTopicInfo_GetFlowControlMaxBytes</code>	Get the volume of pending message bytes at which flow control is enabled for the topic.
<code>tibemsTopicInfo_GetInboundStatistics</code>	Get inbound statistics for this topic.
<code>tibemsTopicInfo_GetMaxBytes</code>	Get the maximum number of bytes of pending messages bound for this topic that the server will store.
<code>tibemsTopicInfo_GetMaxMsgs</code>	Get the maximum number of pending messages bound for the topic that the server will store.
<code>tibemsTopicInfo_GetName</code>	Get the name of this topic.
<code>tibemsTopicInfo_GetOutboundStatistics</code>	Get outbound statistics for this topic.
<code>tibemsTopicInfo_GetOverflowPolicy</code>	Get the overflow policy for this topic.
<code>tibemsTopicInfo_GetPendingMessageCount</code>	Get the total number of pending messages for this topic.
<code>tibemsTopicInfo_GetPendingMessageSize</code>	Get the total size of all pending messages for this topic.
<code>tibemsTopicInfo_GetPendingPersistentMessageCount</code>	Get the total number of pending messages for this topic that were sent persistently.

Function	Description
<a href="#">tibemsTopicInfo_GetPendingPersistentMessageSize</a>	Get the total size of all pending messages for this topic that were sent persistently.
<a href="#">tibemsTopicInfo_GetSubscriberCount</a>	Get the number of subscribers on this topic.
<a href="#">tibemsTopicInfo_GetSubscriptionCount</a>	Get the current number of subscriptions for this topic.

### Related Types

[tibemsStatData](#)

## tibemsTopicInfo\_Create

*Function*

### Purpose

Create a `tibemsTopicInfo` object.

### C Declaration

```
tibems_status tibemsTopicInfo_Create(
    tibemsTopicInfo* topicInfo,
    const char* topicName);
```

### COBOL Call

```
CALL "tibemsTopicInfo_Create"
  USING BY REFERENCE topicInfo,
        BY REFERENCE name,
        RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
<code>topicInfo</code>	Store the new <code>tibemsTopicInfo</code> object in this location.
<code>topicName</code>	Create the <code>tibemsTopicInfo</code> object with this name. The <code>topicName</code> can include wildcards.

### Remarks

This function is used to create a `tibemsTopicInfo` object with the specified name. The `tibemsTopicInfo` object may then be passed as the `tibemsDestinationInfo` argument to the [tibemsAdmin\\_GetConsumers](#) and [tibemsAdmin\\_GetProducerStatistics](#) functions. The name may include wildcards.

For more information on wildcards, see *Wildcards* of the *TIBCO Enterprise Message Service User's Guide*.

### See Also

[tibemsQueueInfo\\_Create](#)

[tibemsAdmin\\_GetConsumers](#)

[tibemsAdmin\\_GetProducerStatistics](#)

## tibemsTopicInfo\_Destroy

*Function*

### Purpose

Destroy a `tibemsTopicInfo` object.

### C Declaration

```
tibems_status tibemsTopicInfo_Destroy(
    tibemsTopicInfo topicInfo);
```

### COBOL Call

```
CALL "tibemsTopicInfo_Destroy"
    USING BY VALUE topicInfo,
         RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
<code>topicInfo</code>	Destroy this <code>tibemsTopicInfo</code> object.

### See Also

[tibemsQueueInfo\\_Destroy](#)

## tibemsTopicInfo\_GetActiveDurableCount

*Function*

### Purpose

Get the current number of active durable subscribers for this topic.

### C Declaration

```
tibems_status tibemsTopicInfo_GetActiveDurableCount(
    tibemsTopicInfo topicInfo,
    tibems_int* count);
```

### COBOL Call

```
CALL "tibemsTopicInfo_GetActiveDurableCount"
    USING BY VALUE topicInfo,
         BY REFERENCE count,
         RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
<code>topicInfo</code>	Get the number of subscribers to this topic.

Parameter	Description
count	Stores the number of active durable subscribers.

### See Also

[tibemsTopicInfo\\_GetSubscriberCount](#)

[tibemsTopicInfo\\_GetSubscriptionCount](#)

[tibemsTopicInfo\\_GetDurableSubscriptionCount](#)

## tibemsTopicInfo\_GetDurableCount

*Function*



This function is deprecated in Software Release 8.0.0, and will no longer be supported in future releases. Applications should be updated to use the [tibemsTopicInfo\\_GetDurableSubscriptionCount](#) function.

### Purpose

Get the current number of durable subscribers for this topic.

### C Declaration

```
tibems_status tibemsTopicInfo_GetDurableCount(
    tibemsTopicInfo topicInfo,
    tibems_int* count);
```

### COBOL Call

```
CALL "tibemsTopicInfo_GetDurableCount"
    USING BY VALUE topicInfo,
         BY REFERENCE count,
         RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
topicInfo	Get the number of subscribers to this topic.
count	Store the number of subscribers in this location.

### See Also

[tibemsTopicInfo\\_GetActiveDurableCount](#)

[tibemsTopicInfo\\_GetSubscriberCount](#)

## tibemsTopicInfo\_GetDurableSubscriptionCount

*Function*

### Purpose

Get the current number of durable subscriptions for this topic.

## C Declaration

```
tibems_status tibemsTopicInfo_GetDurableSubscriptionCount(
    tibemsTopicInfo topicInfo,
    tibems_int* count);
```

## COBOL Call

```
CALL "tibemsTopicInfo_GetDurableSubscriptionCount"
    USING BY VALUE topicInfo,
         BY REFERENCE count,
         RETURNING tibems-status
END-CALL.
```



count has usage pointer.

## Parameters

Parameter	Description
topicInfo	Get the count for this topic.
count	The function stores the number of shared subscriptions in this location.

## See Also

[tibemsTopicInfo\\_GetSubscriptionCount](#)

[tibemsTopicInfo\\_GetActiveDurableCount](#)

[tibemsTopicInfo\\_GetSubscriberCount](#)

## tibemsTopicInfo\_GetFlowControlMaxBytes

*Function*

### Purpose

Get the volume of pending message bytes at which flow control is enabled for the topic.

## C Declaration

```
tibems_status tibemsTopicInfo_GetFlowControlMaxBytes(
    tibemsTopicInfo topicInfo,
    tibems_long* maxBytes);
```

## COBOL Call

```
CALL "tibemsTopicInfo_GetFlowControlMaxBytes"
    USING BY VALUE topicInfo,
         BY REFERENCE maxBytes,
         RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
topicInfo	Get the volume for this topic.

Parameter	Description
maxBytes	Store the volume.  The value stored indicates the volume of pending messages, in bytes, that the server will store for the topic before enabling flow control. A value of 0 indicates that flow control will never be enabled.

**See Also**

[tibemsQueueInfo\\_GetFlowControlMaxBytes](#)

**tibemsTopicInfo\_GetInboundStatistics**

*Function*

**Purpose**

Get inbound statistics for this topic.

**C Declaration**

```
tibems_status tibemsTopicInfo_GetInboundStatistics(
    tibemsTopicInfo topicInfo,
    tibemsStatData* statData);
```

**COBOL Call**

```
CALL "tibemsTopicInfo_GetInboundStatistics"
    USING BY VALUE topicInfo,
         BY REFERENCE statData,
         RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
topicInfo	Get statistics for this topic.
statData	Store statistics.

**Remarks**

This function retrieves the inbound statistics for the topic. Inbound statistics include all messages sent by EMS clients and routed servers.

**See Also**

[tibemsQueueInfo\\_GetInboundStatistics](#)

**tibemsTopicInfo\_GetMaxBytes**

*Function*

**Purpose**

Get the maximum number of bytes of pending messages bound for this topic that the server will store.

**C Declaration**

```
tibems_status tibemsTopicInfo_GetMaxBytes(
    tibemsTopicInfo topicInfo,
    tibems_long* maxBytes);
```

**COBOL Call**

```
CALL "tibemsTopicInfo_GetMaxBytes"
    USING BY VALUE topicInfo,
         BY REFERENCE maxBytes,
         RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
topicInfo	Get the number of bytes for this server.
maxBytes	Stores the number of bytes.

**See Also**

[tibemsQueueInfo\\_GetMaxBytes](#)

**tibemsTopicInfo\_GetMaxMsgs**

*Function*

**Purpose**

Get the maximum number of pending messages bound for the topic that the server will store.

**C Declaration**

```
tibems_status tibemsTopicInfo_GetMaxMsgs(
    tibemsTopicInfo topicInfo,
    tibems_long* maxMsgs);
```

**COBOL Call**

```
CALL "tibemsTopicInfo_GetMaxMsgs"
    USING BY VALUE topicInfo,
         BY REFERENCE maxMsgs,
         RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
topicInfo	Get the number of messages for this topic.
maxMsgs	Store the number of messages.

**See Also**

[tibemsQueueInfo\\_GetMaxMsgs](#)

## tibemsTopicInfo\_GetName

*Function*

### Purpose

Get the name of this topic.

### C Declaration

```
tibems_status tibemsTopicInfo_GetName(
    tibemsTopicInfo topicInfo,
    char* name,
    tibems_int name_len);
```

### COBOL Call

```
CALL "tibemsTopicInfo_GetName"
  USING BY VALUE topicInfo,
        BY REFERENCE name,
        BY VALUE name_len,
        RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
topicInfo	Get the name of this topic.
name	Stores the topic name in this location.
name_len	Length of the name buffer.

### See Also

[tibemsQueueInfo\\_GetName](#)

## tibemsTopicInfo\_GetOutboundStatistics

*Function*

### Purpose

Get outbound statistics for this topic.

### C Declaration

```
tibems_status tibemsTopicInfo_GetOutboundStatistics(
    tibemsTopicInfo topicInfo,
    tibemsStatData* statData);
```

### COBOL Call

```
CALL "tibemsTopicInfo_GetOutboundStatistics"
  USING BY VALUE topicInfo,
        BY REFERENCE statData,
        RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
topicInfo	Get statistics for this topic.
statData	Stores the statistics.

## Remarks

This function retrieves the outbound statistics for the topic. Outbound statistics include all messages delivered by the topic to EMS clients and routed servers.

## See Also

[tibemsQueueInfo\\_GetOutboundStatistics](#)

## tibemsTopicInfo\_GetOverflowPolicy

*Function*

### Purpose

Get the overflow policy for this topic.

### C Declaration

```
tibems_status tibemsTopicInfo_GetOverflowPolicy(
    tibemsTopicInfo topicInfo,
    tibems_int* overflowPolicy);
```

### COBOL Call

```
CALL "tibemsTopicInfo_GetOverflowPolicy"
    USING BY VALUE topicInfo,
         BY REFERENCE overflowPolicy,
         RETURNING tibems-status
END-CALL.
```

## Parameters

Parameter	Description
topicInfo	
overflowPolicy	Stores the overflow policy.

## Remarks

This function retrieves the overflow policy for the queue. Possible values are:

- TIBEMS\_OVERFLOW\_DEFAULT
- TIBEMS\_OVERFLOW\_DISCARD\_OLD
- TIBEMS\_OVERFLOW\_REJECT\_INCOMING

For more information about overflow policies, see the *TIBCO Enterprise Message Service User's Guide*.

**See Also**

[tibemsQueueInfo\\_GetOverflowPolicy](#)

**tibemsTopicInfo\_GetPendingMessageCount**

*Function*

**Purpose**

Get the total number of pending messages for this topic.

**C Declaration**

```
tibems_status tibemsTopicInfo_GetPendingMessageCount(
    tibemsTopicInfo topicInfo,
    tibems_long* count);
```

**COBOL Call**

```
CALL "tibemsTopicInfo_GetPendingMessageCount"
  USING BY VALUE topicInfo,
        BY REFERENCE size,
        RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
topicInfo	Get the number of messages for this topic.
count	Stores the number of messages.

**See Also**

[tibemsQueueInfo\\_GetPendingMessageCount](#)

**tibemsTopicInfo\_GetPendingMessageSize**

*Function*

**Purpose**

Get the total size of all pending messages for this topic.

**C Declaration**

```
tibems_status tibemsTopicInfo_GetPendingMessageSize(
    tibemsTopicInfo topicInfo,
    tibems_long* count);
```

**COBOL Call**

```
CALL "tibemsTopicInfo_GetPendingMessageSize"
  USING BY VALUE topicInfo,
        BY REFERENCE size,
        RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
topicInfo	Get the size of messages for this topic.
count	Stores the size of pending messages.

**See Also**

[tibemsQueueInfo\\_GetPendingMessageSize](#)

**tibemsTopicInfo\_GetPendingPersistentMessageCount**

*Function*

**Purpose**

Get the total number of pending messages for this topic that were sent persistently.

**C Declaration**

```
tibems_status tibemsTopicInfo_GetPendingPersistentMessageCount(
    tibemsTopicInfo topicInfo,
    tibems_long* count);
```

**COBOL Call**

```
CALL "tibemsTopicInfo_GetPendingPersistentMessageCount"
  USING BY VALUE topicInfo,
        BY REFERENCE size,
        RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
topicInfo	Get the number of messages for this topic.
count	Stores the number of messages.

**See Also**

[tibemsQueueInfo\\_GetPendingPersistentMessageCount](#)

**tibemsTopicInfo\_GetPendingPersistentMessageSize**

*Function*

**Purpose**

Get the total size of all pending messages for this topic that were sent persistently.

**C Declaration**

```
tibems_status tibemsTopicInfo_GetPendingPersistentMessageSize(
    tibemsTopicInfo topicInfo,
    tibems_long* size);
```

**COBOL Call**

```
CALL "tibemsTopicInfo_GetPendingPersistentMessageSize"
  USING BY VALUE topicInfo,
        BY REFERENCE size,
        RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
topicInfo	Get the size of messages for this topic.
count	Stores the size of pending messages.

**See Also**

[tibemsQueueInfo\\_GetPendingPersistentMessageSize](#)

**tibemsTopicInfo\_GetSubscriberCount**

*Function*

**Purpose**

Get the number of subscribers on this topic.

**C Declaration**

```
tibems_status tibemsTopicInfo_GetSubscriberCount(
  tibemsTopicInfo topicInfo,
  tibems_int* count);
```

**COBOL Call**

```
CALL "tibemsTopicInfo_GetSubscriberCount"
  USING BY VALUE topicInfo,
        BY REFERENCE count,
        RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
topicInfo	Get the count for this topic.
count	Store the number of active subscribers.

**See Also**

[tibemsTopicInfo\\_GetActiveDurableCount](#)

[tibemsTopicInfo\\_GetSubscriptionCount](#)

[tibemsTopicInfo\\_GetDurableSubscriptionCount](#)

## tibemsTopicInfo\_GetSubscriptionCount

*Function*

### Purpose

Get the current number of subscriptions for this topic.

### C Declaration

```
tibems_status tibemsTopicInfo_GetSubscriptionCount(
    tibemsTopicInfo topicInfo,
    tibems_int* count);
```

### COBOL Call

```
CALL "tibemsTopicInfo_GetSubscriptionCount"
    USING BY VALUE topicInfo,
         BY REFERENCE count,
         RETURNING tibems-status
END-CALL.
```



count has usage pointer.

### Parameters

Parameter	Description
topicInfo	Get the count for this topic.
count	Stores the number of subscriptions.

### See Also

[tibemsTopicInfo\\_GetDurableSubscriptionCount](#)

# Exception

The following topics present exceptions related to EMS.

## tibems\_status

*Type*

### Purpose

Functions return status codes to indicate return conditions.

### Status Codes

Constant	Code	Description
TIBEMS_OK	0	The call completed normally.
TIBEMS_ILLEGAL_STATE	1	A function call or server request occurred in an inappropriate context.  For example, <code>tibemsSession_Commit</code> indicates this status when the session is non-transactional.
TIBEMS_INVALID_CLIENT_ID	2	The provider rejects the connection's client ID.  Setting a connection's client ID to an invalid or duplicate value results in this exception. (A duplicate value is one that is already in use by another connection.)
TIBEMS_INVALID_DESTINATION	3	<code>tibemsd</code> cannot locate the destination.
TIBEMS_INVALID_SELECTOR	4	The client passed a message selector with invalid syntax; see <a href="#">Message Selectors</a> .
TIBEMS_EXCEPTION	5	Non-specific error code.
TIBEMS_SECURITY_EXCEPTION	6	The function cannot complete because of a security restriction.  For example, the provider rejects a user or the user's authentication.
TIBEMS_MSG_EOF	7	The data stream within a message ended unexpectedly.  <code>tibemsBytesMsg</code> contains a stream of bytes. <code>tibemsStreamMsg</code> contains a stream of characters. If any of their read functions detects the end of a stream unexpectedly, it indicates this status.
TIBEMS_MSG_NOT_READABLE	9	Attempt to read from a message in write-only mode.

Constant	Code	Description
TIBEMS_MSG_NOT_WRITEABLE	10	Attempt to write to a message in read-only mode.  See also, <a href="#">tibemsMsg_MakeWriteable</a> .
TIBEMS_SERVER_NOT_CONNECTED	11	<ul style="list-style-type: none"> <li>• An attempt to connect to the server has failed.</li> <li>• The operation requires a server connection, but the program is not connected.</li> </ul>
TIBEMS_SUBJECT_COLLISION	13	The server cannot create a topic or durable because the name is already in use. (Also applies to collisions with external subjects, such as Rendezvous.)
TIBEMS_INVALID_PROTOCOL	15	Cannot create a connection or transaction because the specified protocol does not exist.
TIBEMS_INVALID_HOSTNAME	17	The connection URL includes an invalid hostname, or an attempt to lookup the host address failed.  Host names must be less than 128 characters.
TIBEMS_INVALID_PORT	18	The connection URL includes an invalid port number.
TIBEMS_NO_MEMORY	19	The program exceeded available memory during the call.
TIBEMS_INVALID_ARG	20	The function received an illegal value as an argument.
TIBEMS_SERVER_LIMIT	21	The server has exceeded the maximum number of licensed connections or hosts that it can service.
TIBEMS_NOT_PERMITTED	27	The function call is not permitted (for example, closing a connection within a callback).
TIBEMS_SERVER_RECONNECTED	28	Exception callback handler functions receive this code to indicate that the server has reconnected.  See <a href="#">tibemsExceptionCallback</a>

Constant	Code	Description
TIBEMS_INVALID_NAME	30	In a lookup request, the name has incorrect syntax.  The most common syntax error is a prefix other than <code>tibjmsnaming://</code> (or a misspelling).  See also, <a href="#">tibemsLookupContext</a> .
TIBEMS_INVALID_SIZE	32	An argument is outside the range of valid values.
TIBEMS_NOT_FOUND	35	1. The name lookup repository cannot find a name; the name is not bound. See also, <a href="#">tibemsLookupContext</a>  2. A function that gets a message field or property value cannot find the specified item because the name is not bound in the message.
TIBEMS_CONVERSION_FAILED	38	A datatype conversion failed while parsing a message (converting UTF-8 data to native datatypes).
TIBEMS_INVALID_MSG	42	The message is uninitialized or corrupt.
TIBEMS_INVALID_FIELD	43	The message contains an invalid field. The message might be corrupt.
TIBEMS_CORRUPT_MSG	45	The message is corrupt.
TIBEMS_TIMEOUT	50	The timeout has expired while waiting for a message.  See <a href="#">tibemsMsgConsumer_ReceiveTimeout</a> .
TIBEMS_INTR	51	A blocking operation has been interrupted. See <a href="#">tibemsMsgConsumer_Receive</a> .
TIBEMS_DESTINATION_LIMIT_EXCEEDED	52	A server queue or topic has exceeded its size limit, and cannot add a new message.
TIBEMS_MEM_LIMIT_EXCEEDED	53	The server has exceeded its memory limit.
TIBEMS_USER_INTR	54	IBM z/OS only. A blocking operation has been interrupted. See <a href="#">tibx_MVSConsole_SetConsumer()</a> .

Constant	Code	Description
TIBEMS_INVALID_IO_SOURCE	65	The function detected an invalid I/O source (such as a socket or file).
TIBEMS_OS_ERROR	68	An operating system error occurred during the call.
TIBEMS_INSUFFICIENT_BUFFER	70	The result of the call overflowed the buffer supplied by the program.
TIBEMS_EOF	71	The call detected an unexpected end-of-file.
TIBEMS_INVALID_FILE	72	The function detected an invalid file.
TIBEMS_FILE_NOT_FOUND	73	The specified file does not exist.
TIBEMS_IO_FAILED	74	An operating system I/O call failed.
TIBEMS_ALREADY_EXISTS	91	Cannot create an item that already exists.
TIBEMS_INVALID_CONNECTION	100	The connection is invalid.
TIBEMS_INVALID_SESSION	101	The session is invalid.
TIBEMS_INVALID_CONSUMER	102	The consumer is invalid.
TIBEMS_INVALID_PRODUCER	103	The producer is invalid.
TIBEMS_INVALID_USER	104	The server could not authenticate the user.
TIBEMS_TRANSACTION_FAILED	106	A transaction failed at the server during a commit call.
TIBEMS_TRANSACTION_ROLLBACK	107	Failure during prepare or commit caused automatic rollback of a transaction. This type of rollback can occur during fault tolerance failover.
TIBEMS_TRANSACTION_RETRY	108	A transaction failed during two-phase commit; the program may attempt to commit it again.
TIBEMS_INVALID_XARESOURCE	109	When a session uses an XA transaction manager, the XA resource is the correct locus for all commit and rollback requests. Local commit or rollback calls are not permitted, and indicate this status.

Constant	Code	Description
TIBEMS_FT_SERVER_LACKS_TRANSACTION	110	The producer attempted to send a message immediately after a fault tolerance failover to another server. The new server has no record of the transaction.
TIBEMS_NOT_INITIALIZED	200	Initialization of the tibems library failed. For example, this code could be generated if the library failed to allocate memory while building its basic structures.
SSL		
TIBEMS_INVALID_CERT	150	SSL detected an invalid X.509 certificate.
TIBEMS_INVALID_CERT_NOT_YET	151	SSL detected an X.509 certificate that is not yet valid; that is, the current date is before the first date for which the certificate becomes valid.
TIBEMS_INVALID_CERT_EXPIRED	152	SSL detected an X.509 certificate that is no longer valid; that is, the current date is after the expiration date.
TIBEMS_INVALID_CERT_DATA	153	SSL detected an X.509 certificate containing corrupt data.
TIBEMS_ALGORITHM_ERROR	154	Error loading a cipher suite algorithm.
TIBEMS_SSL_ERROR	155	Generic SSL error code.
TIBEMS_INVALID_PRIVATE_KEY	156	SSL detected a private key that does not match its public key.
TIBEMS_INVALID_ENCODING	157	SSL detected a certificate encoding that it cannot read.
TIBEMS_NOT_ENOUGH_RANDOM	158	SSL lacks sufficient random data to complete an operation securely.
Unimplemented		
TIBEMS_NOT_IMPLEMENTED	255	The function is not implemented.

## **tibemsStatus\_GetText**

Function

### **Purpose**

Get the text string corresponding to a status code.

## C Declaration

```
const char* tibemsStatus_GetText(
    tibems_status status );
```

## COBOL Call

```
CALL "tibemsStatus_GetText"
    USING BY VALUE status
    RETURNING tibems-Pointer
END-CALL.
```

## Parameters

Parameter	Description
status	Get the text corresponding to this status code.

## tibemsErrorContext

*Type*

### Purpose

Enable additional error tracking

### Remarks

The `tibemsErrorContext` objects collect additional error information beyond the status returned by most EMS calls. When a `tibemsErrorContext` is created, EMS records detailed error information and a stack trace for the last error detected inside the EMS client library. Upon encountering an EMS error, the error information is written to the `tibemsErrorContext` object and then cleared at the start of the next public EMS function call.

Because each thread of execution in an application may contain specific error information, `tibemsErrorContext` objects should be created at the start of each thread and then destroyed before exiting the thread. Threads spawned internally by EMS will automatically create `tibemsErrorContext` objects.

Function	Description
<code>tibemsErrorContext_Create</code>	Create a new error context object.
<code>tibemsErrorContext_Close</code>	Close and free memory associated with an error context.
<code>tibemsErrorContext_GetLastErrorString</code>	Retrieve any available detailed error string associated with the last EMS call.
<code>tibemsErrorContext_GetLastErrorStackTrace</code>	Retrieve a stack trace associated with the last EMS call.

## tibemsErrorContext\_Create

*Function*

### Purpose

Create a new error context object.

### C Declaration

```
tibems_status tibemsErrorContext_Create(
    tibemsErrorContext* errorContext);
```

### COBOL Call

```
CALL "tibemsErrorContext_Create"
    USING BY REFERENCE errorContext,
    RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
errorContext	Store the new error context.

### Remarks

Create a new error context object and enables detailed error information and stack tracing for the thread that calls this function.

Return TIBEMS\_OK, TIBEMS\_INVALID\_ARG, or TIBEMS\_NO\_MEMORY.

### See Also

[tibemsErrorContext\\_Close](#)

## tibemsErrorContext\_Close

*Function*

### Purpose

Close and free memory associated with an error context.

### C Declaration

```
tibems_status tibemsErrorContext_Close(
    tibemsErrorContext errorContext);
```

### COBOL Call

```
CALL "tibemsErrorContext_Close"
    USING BY VALUE errorContext,
    RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
errorContext	Close this error context.

**Remarks**

Return TIBEMS\_OK, TIBEMS\_INVALID\_ARG.

**See Also**

[tibemsErrorContext\\_Create](#)

**tibemsErrorContext\_GetLastErrorString****Purpose**

Retrieve any available detailed error string associated with the last EMS call.

**C Declaration**

```
tibems_status tibemsErrorContext_GetLastErrorString(
    tibemsErrorContext errorContext,
    const char** string);
```

**COBOL Call**

```
CALL "tibemsErrorContext_GetLastErrorString"
    USING BY VALUE errorContext,
         BY REFERENCE string,
         RETURNING tibems-status
END-CALL.
```

**Parameters**

Parameter	Description
errorContext	The error context.
string	Location of the detailed error string.

**Remarks**

Error string includes the day and time the error occurred.

Passing NULL for the errorContext parameter will default to the error context of the current thread. This is useful for retrieving information from within a listener or exception callback. This function returns a pointer, not a copy of the error string. If the last call was considered a non-error, an empty string is returned.

Returns TIBEMS\_OK, TIBEMS\_INVALID\_ARG (string is null, or passing an object created in a different thread), TIBEMS\_NOT\_INITIALIZED (no error context for this thread)

**See Also**

[tibemsErrorContext\\_GetLastErrorStackTrace](#)

## tibemsErrorContext\_GetLastErrorStackTrace

### Purpose

Retrieve a stack trace associated with the last EMS call.

### C Declaration

```
tibems_status tibemsErrorContext_GetLastErrorStackTrace(
    tibemsErrorContext errorContext,
    const char** string);
```

### COBOL Call

```
CALL "tibemsErrorContext_GetLastErrorStackTrace"
    USING BY VALUE errorContext,
         BY REFERENCE string,
    RETURNING tibems-status
END-CALL.
```

### Parameters

Parameter	Description
errorContext	The error context.
string	Location of the stack trace string.

### Remarks

Passing NULL for the `errorContext` parameter will default to the error context of the current thread. This is useful for retrieving information from within a listener or exception callback. This function returns a pointer, not a copy of the stack trace string. If the last call was considered a non-error, an empty string is returned.

Return `TIBEMS_OK`, `TIBEMS_INVALID_ARG` (string is null, or passing an object created in a different thread), `TIBEMS_NOT_INITIALIZED` (no error context for this thread)

### See Also

[tibemsErrorContext\\_GetLastErrorString](#)

# IBM z/OS and IBM i

The following topics present items of interest specific to IBM z/OS and IBM i native operating systems.

## IBM EBCDIC Platform Calls

The calls we present on the following pages are implemented only on IBM EBCDIC platforms.

### tibems\_SetCodePages

*Function*

#### Purpose

Set code pages for string conversion on EBCDIC platforms (when non-default code pages are required).

#### C Declaration

```
tibems_status tibems_SetCodePages(
    char* host_codepage,
    char* net_codepage);
```

#### COBOL Call

```
CALL "tibems_SetCodePages"
    USING BY REFERENCE host-codepage,
          BY REFERENCE net-codepage,
          RETURNING TIBEMS-STATUS
END-CALL.
```

#### Parameters

Parameter	Description
host_codepage	Set this code page as the native (EBCDIC) character encoding for the host computer.
net_codepage	Set this code page as the ASCII or UTF-8 character set for the network.

#### String Conversion

EMS software uses the operating system's `iconv()` call to automatically convert strings within messages. Conversion occurs only as needed:

- Programs running in EBCDIC environments represent all strings using an EBCDIC code page (called the *host code page*). Before sending a message, the EMS client library converts its strings to an ASCII or UTF-8 character set (the *network code page*).
- Conversely, when a message arrives from the network, the EMS client library converts its strings to the EBCDIC host code page before presenting the message to the program.

#### Remarks

This call sets the host and network code pages for string conversions in EBCDIC environments.

Call this function when the system code pages differ from the EMS default code pages (see the table of [Default Code Pages](#)). Throughout an enterprise, all sending and receiving programs must use the same code pages.

Both arguments are string names of code pages. To determine valid code page names for your operating system, see documentation from the operating system vendor.

Programs may call this function at most once. The call *must* precede the first call to any message function, and the arrival of the first message from the network.



On IBM i, note that:

- Arguments are 5 character Coded Character Set Identifiers (CCSID) and must be null-terminated.
- Only the first call to this function within an activation group has any affect on the code pages.

### Default Code Pages

To use a default code page, programs may supply NULL for either parameter. Using the default code pages in both parameter positions has the same effect as not calling this function at all.

Default Host Code Page	Default Network Code Page
On z/OS: "IBM-1047"	"ISO8859-1"
On IBM i, the default is the CCSID of the job.	

### See Also

[Strings and Character Encodings](#)

## IBM z/OS Functions

These functions are implemented only on IBM z/OS platforms.

### tibx\_MVSConsole\_SetConsumer

*Function*

#### Purpose

Exit from a blocking listener.

#### C Declaration

```
tibx_MVSConsole_SetConsumer(
    void* pConsole,
    tibemsMsgConsumer tibemsMsgConsumer,
    char* tibems_MVS_BreakFunction);

signed long int tibems_MVS_BreakFunction(
    void* pConsole );
```

#### COBOL Call

```
SET WS-PROCEDURE-PTR TO ENTRY 'tibems_MVS_BreakFunction'

CALL "tibx_MVSConsole_SetConsumer"
    USING BY VALUE pConsole,
          BY VALUE tibemsMsgConsumer,
          BY VALUE pFunction,
          RETURNING tibems-status
END-CALL.
```



pConsole has usage pointer.

Before this call, you must set the entry to the MVS break function.

## Parameters

Parameter	Description
pConsole	Set the consumer of this MVS console object.
tibemsMsgConsumer	Use this message consumer.
pFunction	In COBOL, use this <code>tibems_MVS_BreakFunction</code> function address.
tibems_MVS_BreakFunction	In C, use this break function name.

## Remarks

Programs in single-threaded environments (such as COBOL) need a way to interrupt blocking receive calls (such as `tibemsMsgConsumer_Receive`).

After registering this function in COBOL, a console stop or **shut** command causes the receive call to return with a status code `TIBEMS_USER_INTR` (54).

## See Also

[tibemsMsgConsumer\\_Receive](#)

[tibx\\_MVSConsole\\_Create\(\)](#)

## tibx\_MVSConsole\_Create

*Function*

### Purpose

Create or destroy an MVS console.

### C Declaration

```
signed long int tibx_MVSConsole_Create (
    void** pConsole,
    char** pConsoleMsg,
    Console_Response pCallBack)

signed long int tibx_MVSConsole_Destroy(
    void* pConsole );
```

### COBOL Call

```
CALL "tibx_MVSConsole_Create"
    USING BY REFERENCE pConsole,
          BY REFERENCE pConsoleMsg,
          BY VALUE     TIBEMS-NULLPTR,
          RETURNING    tibems-status
END-CALL.

CALL "tibx_MVSConsole_Destroy"
    USING BY VALUE pConsole,
          RETURNING tibems-status
END-CALL.
```



pConsole and pConsole-Msg have usage pointer.

## Parameters

Parameter	Description
pConsole	The create call stores the MVS console handle in this location. The destroy call destroys this MVS console.
pConsoleMsg	When the return status code is non-zero, the function stores an error message in this location.
pCallback	C programs define this callback function to receive the results of MVS console commands.

## Remarks

Some consumer application programs wait indefinitely for messages to arrive. You can use this function in conjunction with `tibems_MVS_BreakFunction` to arrange console input to such programs, in order to interrupt them from waiting to receive a message, so they can exit cleanly (see [tibx\\_MVSConsole\\_SetConsumer\(\)](#)).

C programs can receive console command results through a callback function. COBOL programs cannot receive console command results, but can react to the MVS stop and shut commands.

## See Also

[tibemsMsgConsumer\\_Receive](#)

[tibx\\_MVSConsole\\_SetConsumer\(\)](#)

## Console\_Response

*Function Type*

### Purpose

Callback function to relay the results of MVS console commands to C programs.

### C Declaration

```
signed long int
Console_Response(
    signed long int rc,
    char* ops_command );
```

### Parameters

Parameter	Description
rc	Receive the return status code from the MVS console.
ops_command	Receive the operator command from the MVS console.
<i>return value</i>	Zero indicates a normal return. All other values indicate a special console command—namely, stop or shut.

**Remarks**

Available only in C for z/OS MVS.

COBOL does not support callback functions.

**See Also**

[tibx\\_MVSConsole\\_Create\(\)](#)

**SSL Implementation on IBM EBCDIC Systems**

On z/OS and IBM i systems, secure connections are created using IBM System SSL. To implement IBM System SSL for TIBCO Enterprise Message Service, you must set the necessary environment variables, and use the API documented in the sections below.

Additionally, the EMS client application must run within the context of a user ID which has the necessary privileges to access the IBM System SSL facility. For details about these requirements, see the section on SSL requirements in the *TIBCO EMS Client for z/OS (MVS) Installation and Reference*, or the *TIBCO EMS Client for IBM i Installation and Reference*.

**IBM System SSL Environment Variables on z/OS**

There are some System SSL environment variables related to tracing and debug messages which must be specified before the EMS library is loaded.



Tracing is supported only on z/OS systems.

These environment variables are discussed in the IBM System Secure Sockets Layer programming guide, but a short summary is provided here:

- Set the file name (USS only) for raw trace records:
  - GSK\_TRACE\_FILE <trace file name>
  - GSK\_TRACE <mask> — specify 255 to trace everything

These two values enable detailed tracing of all GSK calls to a file. The file may subsequently be formatted for viewing using the `gsktrace` command.
- GSK\_HW\_CRYPT0 <crypto mask> — 65535 to enable all, 0 to disable all hardware cryptographic functions.
- GSK\_SSL\_HW\_DETECT\_MESSAGE — if set, causes brief messages regarding the state of the installed hardware to be written to stdout when the application is started.

**Call Summary**

The following table provides a list of the API calls used to configure IBM System SSL.

IBM System SSL API	Use On	Corresponding Open SSL API
<a href="#">tibemsSSL_System_GetTrace</a> <a href="#">tibemsSSL_System_GetDebugTrace</a>	z/OS IBM i	<a href="#">tibemsSSL_GetTrace</a> <a href="#">tibemsSSL_GetDebugTrace</a>
<a href="#">tibemsSSL_System_SetFipsMode</a>	z/OS	IBM System SSL only
<a href="#">tibemsSSL_System_SetTrace</a> <a href="#">tibemsSSL_System_SetDebugTrace</a>	z/OS IBM i	<a href="#">tibemsSSL_SetTrace</a> <a href="#">tibemsSSL_SetDebugTrace</a>

IBM System SSL API	Use On	Corresponding Open SSL API
<code>tibemsSSL_System_Version</code>	z/OS IBM i	<code>tibemsSSL_OpenSSLVersion</code>
<code>tibemsSSLParams_System_Create</code>	z/OS IBM i	<code>tibemsSSLParams_Create</code>
<code>tibemsSSLParams_System_Destroy</code>	z/OS IBM i	<code>tibemsSSLParams_Destroy</code>
<code>tibemsSSLParams_System_SetApplicationId</code>	IBM i	IBM System SSL only
<code>tibemsSSLParams_System_SetAuthOnly</code>	z/OS IBM i	<code>tibemsSSLParams_SetAuthOnly</code>
<code>tibemsSSLParams_System_SetCiphers</code>	z/OS IBM i	<code>tibemsSSLParams_SetCiphers</code>
<code>tibemsSSLParams_System_SetEnableTLS1</code>	z/OS IBM i	IBM System SSL only
<code>tibemsSSLParams_System_SetEnableTLS11</code>	z/OS IBM i	IBM System SSL only
<code>tibemsSSLParams_System_SetEnableTLS12</code>	z/OS IBM i	IBM System SSL only
<code>tibemsSSLParams_System_SetExpectedHostName</code>	z/OS IBM i	<code>tibemsSSLParams_SetExpectedHostName</code>
<code>tibemsSSLParams_System_SetKeyRingFile</code>	z/OS IBM i	IBM System SSL only
<code>tibemsSSLParams_System_SetLabel</code>	z/OS IBM i	IBM System SSL only
<code>tibemsSSLParams_System_SetLdapServerPassword</code>	z/OS	IBM System SSL only
<code>tibemsSSLParams_System_SetLdapServerUrl</code>	z/OS	IBM System SSL only
<code>tibemsSSLParams_System_SetLdapServerUserid</code>	z/OS	IBM System SSL only
<code>tibemsSSLParams_System_SetVerifyHostName</code>	z/OS IBM i	<code>tibemsSSLParams_SetVerifyHost</code>

## **tibemsSSL\_System\_GetTrace**

*Function*

### **Purpose**

Determine whether SSL tracing is enabled.

**C Declaration**

```
tibems_bool tibemsSSL_System_GetTrace(void);
tibems_bool tibemsSSL_System_GetDebugTrace(void);
```

**COBOL Call**

```
CALL "tibemsSSL_System_GetTrace"
      RETURNING value-Boolean
END-CALL.

CALL "tibemsSSL_System_GetDebugTrace"
      RETURNING value-Boolean
END-CALL.
```

**Remarks**

Two levels of SSL tracing are available—regular tracing and debug tracing (more detailed).

If tracing is enabled, these calls return `TIBEMS_TRUE`.

If tracing is disabled, they return `TIBEMS_FALSE`.

**See Also**

[tibemsSSL\\_GetTrace](#)

**tibemsSSL\_System\_SetFipsMode**

*Function*

**Purpose**

Enable or disable FIPS 140-2 mode.

**C Declaration**

```
tibems_status tibemsSSL_System_SetFipsMode(
    tibems_bool enabled)
```

**COBOL Call**

```
CALL "tibemsSSLParams_System_SetFipsMode"
      USING BY VALUE fipsEnabled,
      RETURNING tibems-status
END-CALL.
```

**IBM i**

This function is not supported on IBM i systems.

**Parameters**

Parameter	Description
fipsEnabled	When <code>TIBEMS_TRUE</code> , FIPS mode is enabled. When <code>TIBEMS_FALSE</code> , FIPS mode is disabled.

**Remarks**

This call, while closely associated with the SSL parameters calls, does not actually set or reset a parameter value. Rather, it enables at an application-wide scope the use (or disuse) of FIPS mode.

This call must be made before any other System SSL related calls and before instantiating an SSL connection factory.

Returns TIBEMS\_OK if it succeeds, otherwise TIBEMS\_SSL\_ERROR.

## tibemsSSL\_System\_SetTrace

*Function*

### Purpose

Enable or disable trace messages on the creation of TIBCO Enterprise Message Service connections. These messages indicate the status of various stages of connection creation.

### C Declaration

```
void tibemsSSL_System_SetTrace(
    tibems_bool trace );

void tibemsSSL_System_SetDebugTrace(
    tibems_bool trace );
```

### COBOL Call

```
CALL "tibemsSSL_System_SetTrace"
    USING BY VALUE trace
END-CALL.

CALL "tibemsSSL_System_SetDebugTrace"
    USING BY VALUE trace
END-CALL.
```

### Parameters

Parameter	Description
trace	TIBEMS_TRUE enables tracing. TIBEMS_FALSE disables tracing.

### Remarks

Two levels of SSL tracing are available—regular tracing and debug tracing (more detailed). Trace messages are written to standard output while connecting or reconnecting to the server. These messages can be helpful in diagnosing a failure to connect.

### See Also

[tibemsSSL\\_SetTrace](#)

## tibemsSSL\_System\_Version

*Function*

### Purpose

Get a string representing the IBM System SSL version number.

### C Declaration

```
const char* tibemsSSL_System_Version(
    char* buffer,
    tibems_int buf_size );
```

**COBOL Call**

```
MOVE LENGTH OF buffer TO buf-size.
CALL "tibemsSSL_System_Version"
    USING BY REFERENCE buffer,
          BY VALUE buf-size,
          RETURNING value-Pointer
END-CALL.
```



buffer has usage pointer.

**Parameters**

Parameter	Description
buffer	Copy the version string in this buffer.
buf_size	Length (in bytes) of the buffer.

**Remarks**

On z/OS, the version string has the format *major.minor.update*.

On IBM i, version number information is not available. A string representing the underlying System SSL programming interface is copied to the buffer.

A null character terminates the version string.

**See Also**

[tibemsSSL\\_OpenSSLVersion](#)

**tibemsSSLParams\_System\_Create**

*Function*

**Purpose**

Create a new IBM System SSL parameter object.

**C Declaration**

```
tibemsSSLParams tibemsSSLParams_System_Create(void);
```

**COBOL Call**

```
CALL "tibemsSSLParams_System_Create"
    RETURNING SSLParams
END-CALL.
```



SSLParams has usage pointer.

**Remarks**

Storage is allocated to contain the values associated with various SSL parameters. This call must precede any call that requires an SSL parameters object as an argument.

**See Also**

[tibemsSSLParams\\_Create](#)

## tibemsSSLParams\_System\_Destroy

*Function*

### Purpose

Destroy an IBM System SSL parameter object.

### C Declaration

```
void tibemsSSLParams_System_Destroy(  
    tibemsSSLParams SSLParams );
```

### COBOL Call

```
CALL "tibemsSSLParams_System_Destroy"  
    USING BY VALUE SSLParams  
END-CALL.
```



SSLParams has usage pointer.

### Parameters

Parameter	Description
SSLParams	Destroy this SSL parameter object.

### Remarks

Storage related to the parameters object and all individual parameters therein is released.

### See Also

[tibemsSSLParams\\_Destroy](#)

## tibemsSSLParams\_System\_SetApplicationId

*Function*

### Purpose

Set the application ID to be used by this application.

### C Declaration

```
tibems_status tibemsSSLParams_System_SetApplicationId(  
    tibemsSSLParams params,  
    const char* application_id );
```

### COBOL Call

```
CALL "tibemsSSLParams_System_SetApplicationId"  
    USING BY VALUE params,  
          BY REFERENCE application-id,  
          RETURNING tibems-status  
END-CALL.
```



application-id has usage pointer.

## IBM z/OS

This function is not supported on IBM z/OS systems.

### Parameters

Parameter	Description
params	Set the value in this SSL parameter object.
application_id	The application ID for the application definition.

### Remarks

The IBM i Digital Certificate Manager may be used to create a client application definition and assign it a certificate to be used during an SSL handshake. Part of the application definition is the application ID, which uniquely identifies the application definition. More information about application definitions may be found in the IBM i Information Center, Security, Digital Certificate Manager.

This is a global parameter. As a result, the first connection made by the application establishes the application ID to be used by all other connections within this address space. Any attempt to specify a different application ID for subsequent connections will be ignored.

Because a connection requires either an application ID *or* a certificate store and label—but not both—setting this parameter causes any parameters set with [tibemsSSLParams\\_System\\_SetKeyRingFile](#) and [tibemsSSLParams\\_System\\_SetLabel](#) to be ignored.

Returns TIBEMS\_OK if it succeeds, otherwise TIBEMS\_SSL\_ERROR.

### See Also

[tibemsSSLParams\\_System\\_SetKeyRingFile](#)

[tibemsSSLParams\\_System\\_SetLabel](#)

## tibemsSSLParams\_System\_SetAuthOnly

*Function*

### Purpose

Limit the use of IBM System SSL to improve performance.

### C Declaration

```
tibems_status tibemsSSLParams_System_SetAuthOnly(
    tibemsSSLParams SSLParams,
    tibems_bool auth_only );
```

### COBOL Call

```
CALL "tibemsSSLParams_System_SetAuthOnly"
    USING BY VALUE SSLParams,
          BY VALUE auth_only,
          RETURNING tibems-status
END-CALL.
```



SSLParams has usage pointer.

## Parameters

Parameter	Description
SSLParams	Set the value in this SSL parameter object.
auth_only	TIBEMS_TRUE instructs the SSL parameter object to request a connection that uses SSL only for authentication.  TIBEMS_FALSE instructs the SSL parameter object to request a connection that uses SSL to secure all data.

## Remarks

This parameter is connection-specific and can be specified for each connection.

For background information, see [SSL Authentication Only](#) in *TIBCO Enterprise Message Service User's Guide*.

## See Also

[tibemsSSLParams\\_SetAuthOnly](#)

## tibemsSSLParams\_System\_SetCiphers

*Function*

### Purpose

Set the cipher suites for IBM System SSL connections.

### C Declaration

```
tibems_status tibemsSSLParams_System_SetCiphers(
    tibemsSSLParams SSLParams,
    const char* ciphers );
```

### COBOL Call

```
CALL "tibemsSSLParams_System_SetCiphers"
    USING BY VALUE SSLParams,
         BY REFERENCE ciphers,
         RETURNING tibems-status
END-CALL.
```



SSLParams has usage pointer.

## Parameters

Parameter	Description
SSLParams	Set the value in this SSL parameter object.

Parameter	Description
ciphers	<p>Specify the cipher suites that the client can use. Ciphers can be specified as a series of two or four character codes, or a series of short name string values, depending on the platform.</p> <p>The ciphers provided should conform to the cipher suite specifications for IBM System SSL:</p> <ul style="list-style-type: none"> <li>For a detailed description of these ciphers on z/OS, see the IBM Cryptographic Services, System Secure Sockets Layer Programming bookshelf.</li> <li>For a description of the ciphers on IBM i, refer to the IBM i Information Center, Communications, Socket Programming, Advanced socket concepts, Secure Sockets, Global Security Kit (GSKit) APIs.</li> <li>Note that on IBM i the system values QSSLCSL, QSSLCSLCTL, and QSSLPCL control the ciphers and protocols that are supported.</li> </ul> <p>Supported ciphers are listed below.</p>

### Remarks

This parameter is connection-specific and can be specified for each connection. On z/OS, only the indicated CIPHERS are allowed in FIPS mode.

The following table lists ciphers which have been tested. However, many factors can affect the list of ciphers which work on a given site, so your list may be larger or smaller than this one.

2-Char Code	4-Char Code	Short Name	Description	Use On	FIPS
05	0005	TLS_RSA_WITH_RC4_128_SHA	128-bit RC4 encryption with SHA-1 message authentication and RSA key exchange.	z/OS IBM i	No
0A	000A	TLS_RSA_WITH_3DES_EDE_CBC_SHA	168-bit Triple DES encryption with SHA-1 message authentication and RSA key exchange.	z/OS IBM i	Yes
16	0016	TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	168-bit Triple DES encryption with SHA-1 message authentication and ephemeral Diffie-Hellman key exchange signed with an RSA certificate.	z/OS	Yes
2F	002F	TLS_RSA_WITH_AES_128_CBC_SHA	128-bit AES encryption with SHA-1 message authentication and RSA key exchange.	z/OS IBM i	Yes

2-Char Code	4-Char Code	Short Name	Description	Use On	FIPS
33	0033	TLS_DHE_RSA_WITH_AES_128_CBC_SHA	128-bit AES encryption with SHA-1 message authentication and ephemeral Diffie-Hellman key exchange signed with an RSA certificate.	z/OS	Yes
35	0035	TLS_RSA_WITH_AES_256_CBC_SHA	256-bit AES encryption with SHA-1 message authentication and RSA key exchange.	z/OS IBM i	Yes
39	0039	TLS_DHE_RSA_WITH_AES_256_CBC_SHA	256-bit AES encryption with SHA-1 message authentication and ephemeral Diffie-Hellman key exchange signed with an RSA certificate.	z/OS	Yes
3C	003C	TLS_RSA_WITH_AES_128_CBC_SHA256	128-bit AES encryption with SHA-256 message authentication and RSA key exchange.	z/OS	Yes
3D	003D	TLS_RSA_WITH_AES_256_CBC_SHA256	256-bit AES encryption with SHA-256 message authentication and RSA key exchange.	z/OS	Yes
67	0067	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	128-bit AES encryption with SHA-256 message authentication and ephemeral Diffie-Hellman key exchange signed with an RSA certificate.	z/OS	Yes
6B	006B	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	256-bit AES encryption with SHA-256 message authentication and ephemeral Diffie-Hellman key exchange signed with an RSA certificate.	z/OS	Yes
9C	009C	TLS_RSA_WITH_AES_128_GCM_SHA256	128-bit AES in Galois Counter Mode encryption with 128-bit AEAD authentication and RSA key exchange z/OS.	z/OS	Yes

2-Char Code	4-Char Code	Short Name	Description	Use On	FIPS
9D	009D	TLS_RSA_WITH_AES_256_GCM_SHA384	256-bit AES in Galois Counter Mode encryption with 128-bit AEAD authentication and RSA key exchange z/OS.	z/OS	Yes
9E	009E	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	128-bit AES in Galois Counter Mode encryption with 128-bit AEAD authentication and ephemeral Diffie-Hellman key exchange signed with an RSA certificate.	z/OS	Yes
9F	009F	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	256-bit AES in Galois Counter Mode encryption with 128-bit AEAD authentication and ephemeral Diffie-Hellman key exchange signed with an RSA certificate.	z/OS	Yes
	C011	TLS_ECDHE_RSA_WITH_RC4_128_SHA	128-bit RC4 encryption with SHA-1 message authentication and ephemeral ECDH key exchange signed with an RSA certificate.	z/OS	No
	C012	TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	168-bit Triple DES encryption with SHA-1 message authentication and ephemeral ECDH key exchange signed with an RSA certificate.	z/OS	Yes
	C013	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	128-bit AES encryption with SHA-1 message authentication and ephemeral ECDH key exchange signed with an RSA certificate.	z/OS	Yes
	C014	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	256-bit AES encryption with SHA-1 message authentication and ephemeral ECDH key exchange signed with an RSA certificate.	z/OS	Yes

2-Char Code	4-Char Code	Short Name	Description	Use On	FIPS
	C027	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	128-bit AES encryption with SHA-256 message authentication and ephemeral ECDH key exchange signed with an RSA certificate.	z/OS	Yes
	C028	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	256-bit AES encryption with SHA-384 message authentication and ephemeral ECDH key exchange signed with an RSA certificate.	z/OS	Yes
	C02F	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	128-bit AES in Galois Counter Mode encryption with 128-bit AEAD message authentication and ephemeral ECDH key exchange signed with an RSA certificate.	z/OS	Yes
	C030	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	256-bit AES in Galois Counter Mode encryption with 128-bit AEAD message authentication and ephemeral ECDH key exchange signed with an RSA certificate.	z/OS	Yes

On both z/OS and IBM i the ciphers may be specified as a string of two-character codes with no spaces or other delimiters. For example "2F0535".

On z/OS the ciphers may also be specified as a string of four-character codes with no spaces or other delimiters. For example "C012003D003C".

On IBM i the ciphers may alternately be specified as a string of comma-delimited string values containing no spaces. For example,

"TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA,TLS\_RSA\_WITH\_RC4\_128\_SHA,TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA".

### See Also

[tibemsSSLParams\\_SetCiphers](#)

## tibemsSSLParams\_System\_SetEnableTLS1

*Function*

### Purpose

Enable or disable the TLSV1 protocol.

## C Declaration

```
tibems_status tibemsSSLParams_System_SetEnableTLS1(
    tibemsSSLParams params,
    tibems_bool TLS1enabled)
```

## COBOL Call

```
CALL "tibemsSSLParams_System_SetEnableTLS1"
    USING BY VALUE params,
          BY VALUE TLS1enabled,
          RETURNING tibems_status
END-CALL.
```



params has usage pointer.

## Parameters

Parameter	Description
params	The SSL parameter object.
TLS1enabled	Specify TIBEMS_TRUE or TIBEMS_FALSE to enable or disable the TLSV1 protocol.  This protocol is enabled by default.

## Remarks

Enable or disable the TLSV1 protocol. This parameter is connection-specific and can be specified for each connection.

Returns TIBEMS\_OK if it succeeds, otherwise TIBEMS\_SSL\_ERROR.

## tibemsSSLParams\_System\_SetEnableTLS11

*Function*

### Purpose

Enable or disable the TLSV1.1 protocol.

## C Declaration

```
tibems_status tibemsSSLParams_System_SetEnableTLS11(
    tibemsSSLParams params,
    tibems_bool TLS11enabled);
```

## COBOL Call

```
CALL "tibemsSSLParams_System_SetEnableTLS11"
    USING BY VALUE params,
          BY VALUE TLS11enabled,
          RETURNING tibems_status
END-CALL.
```



SSLParams has a usage pointer.

## Parameters

Parameter	Description
SSLParams	Set the value in this SSL parameter object.
TLS11enabled	Specify TIBEMS_TRUE or TIBEMS_FALSE to enable or disable the SSLV1.1 protocol.  This protocol is enabled by default.

## Remarks

Enable or disable the TLS1.1 protocol. This parameter is connection-specific and can be specified for each connection.

Returns TIBEMS\_OK if it succeeds, otherwise TIBEMS\_SSL\_ERROR.

## tibemsSSLParams\_System\_SetEnableTLS12

*Function*

### Purpose

Enable or disable the TLSV1.2 protocol.

### C Declaration

```
tibems_status tibemsSSLParams_System_SetEnableTLS12(
    tibemsSSLParams params,
    tibems_bool TLS12enabled);
```

### COBOL Call

```
CALL "tibemsSSLParams_System_SetEnableTLS12"
    USING BY VALUE params,
          BY VALUE TLS12enabled,
          RETURNING tibems_status
END-CALL.
```



SSLParams has a usage pointer.

## Parameters

Parameter	Description
SSLParams	Set the value in this SSL parameter object.
TLS12enabled	Specify TIBEMS_TRUE or TIBEMS_FALSE to enable or disable the SSLV1.2 protocol.  This protocol is enabled by default.

## Remarks

Enable or disable the TLS1.2 protocol. This parameter is connection-specific and can be specified for each connection.

Returns TIBEMS\_OK if it succeeds, otherwise TIBEMS\_SSL\_ERROR.

## tibemsSSLParams\_System\_SetExpectedHostName

*Function*

### Purpose

Set the expected host name.

### C Declaration

```
tibems_status tibemsSSLParams_System_SetExpectedHostName(
    tibemsSSLParams SSLParams,
    const char* expected_hostname );
```

### COBOL Call

```
CALL "tibemsSSLParams_System_SetExpectedHostName"
    USING BY VALUE SSLParams,
         BY REFERENCE expected-hostname,
         RETURNING tibems-status
END-CALL.
```



SSLParams and expected\_hostname have usage pointer.

### Parameters

Parameter	Description
SSLParams	Set the value in this SSL parameter object.
expected_hostname	Use this value.

### Remarks

This parameter applies when establishing an SSL connection to the EMS server. If host name verification is enabled, the EMS client compares the name specified in this call to the CN of the certificate presented by the server during the initial SSL handshake. If they are not the same, the connection fails.

This parameter is connection-specific and can be specified for each connection.

### See Also

[tibemsSSLParams\\_SetExpectedHostName](#)

## tibemsSSLParams\_System\_SetKeyRingFile

*Function*

### Purpose

Set the SAF key ring to be used by this application.

### C Declaration

```
tibems_status tibemsSSLParams_System_SetKeyRingFile(
    tibemsSSLParams params,
    const char* keyring_file)
```

## COBOL Call

```
CALL "tibemsSSLParams_System_SetKeyRingFile"
    USING BY VALUE params,
          BY REFERENCE keyring_file,
          RETURNING tibems-status
END-CALL.
```



params and keyring\_file have usage pointer.

## Parameters

Parameter	Description
params	The SSL parameter object.
keyring_file	On z/OS, the name of the SAF key ring on which the certificates for this application reside.  On IBM i, the name of the certificate store defined within the Digital Certificate Manager. For example, *SYSTEM.

## Remarks

This is a global parameter which means that the first connection made by the application will establish the key ring or certificate store to be used by all other connections within this address space. Any attempt to specify a different key ring for subsequent connections will be ignored.

On z/OS, this parameter is required to make a connection. On IBM i this parameter is required to make a connection if no application ID parameter has been set. If an application ID is set, the key ring file parameter is ignored.

Returns TIBEMS\_OK if it succeeds, otherwise TIBEMS\_SSL\_ERROR.

## See Also

[tibemsSSLParams\\_System\\_SetLabel](#)

[tibemsSSLParams\\_System\\_SetApplicationId](#)

## tibemsSSLParams\_System\_SetLabel

*Function*

### Purpose

Set the certificate to be used to make the connection.

### C Declaration

```
tibems_status tibemsSSLParams_System_SetLabel(
    tibemsSSLParams params,
    const char* keyring_label)
```

## COBOL Call

```
CALL "tibemsSSLParams_System_SetLabel"
    USING BY VALUE SSLParams,
          BY REFERENCE keyring_label,
          RETURNING tibems-status
END-CALL.
```



SSLParams and keyring\_label have usage pointer.

### Parameters

Parameter	Description
params	The SSL parameter object.
keyring_label	The name of the certificate label on the chosen SAF key ring or certificate store to use when creating the connection.

### Remarks

This is a connection-specific parameter and can be separately specified for each connection. The label parameter is optional. If it is not specified, the system uses the default certificate on the key ring. If there is no default certificate, the connection fails.

On IBM i, this parameter setting is ignored if the application ID parameter has been set.

Returns TIBEMS\_OK if it succeeds, otherwise TIBEMS\_SSL\_ERROR.

### See Also

[tibemsSSLParams\\_System\\_SetKeyRingFile](#)

[tibemsSSLParams\\_System\\_SetApplicationId](#)

## tibemsSSLParams\_System\_SetLdapServerPassword

*Function*

### Purpose

Set the password to be used to access LDAP for certificates and CRLs.

### C Declaration

```
tibems_status tibemsSSLParams_System_SetLdapServerPassword(
    tibemsSSLParams params,
    const char* password)
```

### COBOL Call

```
CALL "tibemsSSLParams_System_SetLdapServerPassword"
    USING BY VALUE SSLParams,
         BY REFERENCE password,
         RETURNING tibems-status
END-CALL.
```



SSLParams and password have usage pointer.

### IBM i

This function is not supported on IBM i systems.

### Parameters

Parameter	Description
params	SSL parameter object.

Parameter	Description
password	LDAP server authenticating password.

### Remarks

Once LDAP parameters are established, these parameters remain in effect for the life of the client application, and cannot be overridden by subsequent specifications. The first connection to be established sets the LDAP parameters for all subsequent connections.

Returns TIBEMS\_OK if it succeeds, otherwise TIBEMS\_SSL\_ERROR.

### See Also

[tibemsSSLParams\\_System\\_SetLdapServerUrl](#)

[tibemsSSLParams\\_System\\_SetLdapServerUserid](#)

## tibemsSSLParams\_System\_SetLdapServerUrl

*Function*

### Purpose

Set the server URLs to be used to access LDAP for certificates and CRLs.

### C Declaration

```
tibems_status tibemsSSLParams_System_SetLdapServerUrl(
    tibemsSSLParams params,
    const char* urls)
```

### COBOL Call

```
CALL "tibemsSSLParams_System_SetLdapServerUrl"
    USING BY VALUE SSLParams,
         BY REFERENCE urls,
         RETURNING tibems-status
END-CALL.
```



SSLParams and password have usage pointer.

### IBM i

This function is not supported on IBM i systems.

### Parameters

Parameter	Description
params	SSL parameter object.
urls	URLs of the LDAP servers to be used for obtaining trusted certificates and CRLs. The format of the URL is:  <i>server[:port] server2[:port2]</i>  where <i>server</i> is the EMS server name or IP address and <i>port</i> is the server port number. If the server runs on the standard LDAP port, 389, specifying the <i>port</i> parameter is optional. Specify multiple servers in a space-delimited string.

**Remarks**

Once LDAP parameters are established, these parameters remain in effect for the life of the client application, and cannot be overridden by subsequent specifications. The first connection to be established sets the LDAP parameters for all subsequent connections.

Returns TIBEMS\_OK if it succeeds, otherwise TIBEMS\_SSL\_ERROR.

**See Also**

[tibemsSSLParams\\_System\\_SetLdapServerPassword](#)

[tibemsSSLParams\\_System\\_SetLdapServerUserid](#)

**tibemsSSLParams\_System\_SetLdapServerUserid**

*Function*

**Purpose**

Set the user ID to be used to access LDAP for certificates and CRLs.

**C Declaration**

```
tibems_status tibemsSSLParams_System_SetLdapServerUserid(
    tibemsSSLParams params,
    const char* userid)
```

**COBOL Call**

```
CALL "tibemsSSLParams_System_SetLdapServerUserid"
    USING BY VALUE SSLParams,
         BY REFERENCE userid,
         RETURNING tibems-status
END-CALL.
```



SSLParams and password have usage pointer.

**IBM i**

This function is not supported on IBM i systems.

**Parameters**

Parameter	Description
params	The SSL parameter object.
userid	The user ID to be used when authenticating with the LDAP server.

**Remarks**

Once the LDAP parameters are established, these parameters remain in effect for the life of the client application, and cannot be overridden by subsequent specifications. That is, the first connection to be established sets the LDAP parameters for all subsequent connections.

Returns TIBEMS\_OK if it succeeds, otherwise TIBEMS\_SSL\_ERROR.

**See Also**

[tibemsSSLParams\\_System\\_SetLdapServerPassword](#)

[tibemsSSLParams\\_System\\_SetLdapServerUrl](#)

## tibemsSSLParams\_System\_SetVerifyHostName

*Function*

### Purpose

Enable or disable the verification of the server's hostname during connection creation.

### C Declaration

```
tibems_status tibemsSSLParams_System_SetVerifyHostName(
    tibemsSSLParams SSLParams,
    tibems_bool verify );
```

### COBOL Call

```
CALL "tibemsSSLParams_System_SetVerifyHostName"
    USING BY VALUE SSLParams,
          BY VALUE verify,
          RETURNING tibems-status
END-CALL.
```



SSLParams has usage pointer.

### Parameters

Parameter	Description
SSLParams	Set the value in this SSL parameter object.
verify	TIBEMS_TRUE enables verification. TIBEMS_FALSE disables verification.

### Remarks

tibemsSSLParams\_System\_SetVerifyHostName enables checking the server's actual host name against an expected server host name. If no "expected hostname" has been established, the server's fully qualified hostname is compared with the CN in the certificate presented by the server during the initial handshake. If they are not equal, the connection fails.

This parameter is connection-specific and can be specified for each connection.

This verification action is enabled by default (unless a program explicitly disables it).

### See Also

[tibemsSSLParams\\_System\\_SetExpectedHostName](#)