

TIBCO Enterprise Message Service™

C and COBOL Reference

*System Release 4.3
February 2006*

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 *TIBCO ENTERPRISE MESSAGE SERVICE USER'S GUIDE*). 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 contains confidential information that 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.

TIB, TIBCO, Information Bus, The Power of Now, TIBCO Adapter, Rendezvous are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

EJB, J2EE, JMS and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. 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.

Copyright © 1999–2006 TIBCO Software Inc. ALL RIGHTS RESERVED.

TIBCO Software Inc. Confidential Information

Contents

Tables	xiii
Preface	xv
Related Documentation	xvi
TIBCO Enterprise Message Service Documentation	xvi
Other TIBCO Product Documentation	xvi
Third Party Documentation	xvi
How to Contact TIBCO Customer Support	xviii
Chapter 1 Introduction	1
Overview	2
Excluded Features and Restrictions	3
Strings and Character Encodings	4
IBM Mainframe—COBOL and C	4
Chapter 2 Programmer's Checklist	5
Link These Library Files	6
32-Bit UNIX	6
64-Bit UNIX	6
Microsoft Windows	7
OpenVMS	9
Chapter 3 Messages	11
Parts of a Message	12
Body Types	13
Headers	14
Properties	18
Setting Message Properties	18
EMS Properties	18
JMS Properties	20
Message Selectors	21
Data Type Conversion	24
tibemsMsg	25
tibemsMsg_Acknowledge	28

tibemsMsg_ClearBody	30
tibemsMsg_ClearProperties	31
tibemsMsg_Create	32
tibemsMsg_CreateCopy	33
tibemsMsg_CreateFromBytes	34
tibemsMsg_Destroy	35
tibemsMsg_GetAsBytes	36
tibemsMsg_GetAsBytesCopy	38
tibemsMsg_GetBodyType	40
tibemsMsg_GetByteSize	41
tibemsMsg_GetCorrelationID	42
tibemsMsg_GetDeliveryMode	44
tibemsMsg_GetDestination	45
tibemsMsg_GetEncoding	46
tibemsMsg_GetExpiration	47
tibemsMsg_GetMessageID	48
tibemsMsg_GetPriority	49
tibemsMsg_GetPropertyNamees	50
tibemsMsg_GetRedelivered	51
tibemsMsg_GetReplyTo	52
tibemsMsg_GetTimestamp	53
tibemsMsg_GetType	54
tibemsMsg_MakeWriteable	55
tibemsMsg_Print	56
tibemsMsg—Properties Get	57
tibemsMsg—Properties Set	60
tibemsMsg_PropertyExists	63
tibemsMsg_SetCorrelationID	64
tibemsMsg_SetDeliveryMode	66
tibemsMsg_SetDestination	67
tibemsMsg_SetEncoding	68
tibemsMsg_SetExpiration	69
tibemsMsg_SetMessageID	70
tibemsMsg_SetPriority	71
tibemsMsg_SetRedelivered	72
tibemsMsg_SetReplyTo	73
tibemsMsg_SetTimestamp	74
tibemsMsg_SetType	75
tibemsBytesMsg	76
tibemsBytesMsg_Create	77
tibemsBytesMsg_GetBodyLength	78
tibemsBytesMsg_GetBytes	79
tibemsBytesMsg—Read	80
tibemsBytesMsg_ReadBytes	84
tibemsBytesMsg_Reset	85

tibemsBytesMsg_SetBytes	86
tibemsBytesMsg—Write	87
tibemsBytesMsg_WriteBytes	90
tibemsMapMsg	91
tibemsMapMsg_Create	92
tibemsMapMsg—Get	93
tibemsMapMsg_GetMapNames	97
tibemsMapMsg_ItemExists	98
tibemsMapMsg—Set	99
tibemsMapMsg_SetBytes	104
tibemsObjectMsg	106
tibemsObjectMsg_Create	107
tibemsObjectMsg_GetObjectBytes	108
tibemsObjectMsg_SetObjectBytes	109
tibemsStreamMsg	110
tibemsStreamMsg_Create	112
tibemsStreamMsg_FreeField	113
tibemsStreamMsg—Read	114
tibemsStreamMsg_ReadBytes	117
tibemsStreamMsg_ReadField	118
tibemsStreamMsg_Reset	119
tibemsStreamMsg—Write	120
tibemsStreamMsg_WriteBytes	124
tibemsTextMsg	125
tibemsTextMsg_Create	126
tibemsTextMsg_GetText	127
tibemsTextMsg_SetText	128
tibemsData	129
tibemsDeliveryMode	130
tibemsMsgEnum	131
tibemsMsgEnum_Destroy	132
tibemsMsgEnum_GetNextName	133
tibemsMsgField	134
tibemsMsgField_Print	137
tibemsMsgType	138
Chapter 4 Destination	139
Destination Overview	140
tibemsDestinationType	143
tibemsDestination	144
tibemsDestination_Copy	145

tibemsDestination_Create	146
tibemsDestination_Destroy	147
tibemsDestination_GetName	148
tibemsDestination_GetType	149
tibemsQueue	150
tibemsQueue_Create	151
tibemsQueue_Destroy	152
tibemsQueue_GetQueueName	153
tibemsTemporaryQueue	154
tibemsTemporaryTopic	155
tibemsTopic	156
tibemsTopic_Create	157
tibemsTopic_Destroy	158
tibemsTopic_GetTopicName	159
Chapter 5 Consumer	161
tibemsMsgConsumer	162
tibemsMsgConsumer_Close	163
tibemsMsgConsumer_GetMsgListener	164
tibemsMsgConsumer_GetMsgSelector	165
tibemsMsgConsumer_Receive	166
tibemsMsgConsumer_ReceiveNoWait	167
tibemsMsgConsumer_ReceiveTimeout	168
tibemsMsgConsumer_SetMsgListener	170
tibemsQueueReceiver	171
tibemsQueueReceiver_GetQueue	172
tibemsTopicSubscriber	173
tibemsTopicSubscriber_GetNoLocal	174
tibemsTopicSubscriber_GetTopic	175
tibemsMsgCallback	176
Chapter 6 Producer	177
tibemsMsgProducer	178
tibemsMsgProducer_Close	180
tibemsMsgProducer_GetDeliveryMode	181
tibemsMsgProducer_GetDisableMessageID	182
tibemsMsgProducer_GetDisableMessageTimestamp	183
tibemsMsgProducer_GetPriority	184
tibemsMsgProducer_GetTimeToLive	185
tibemsMsgProducer_Send	186
tibemsMsgProducer_SetDeliveryMode	189
tibemsMsgProducer_SetDisableMessageID	190

tibemsMsgProducer_SetDisableMessageTimestamp	191
tibemsMsgProducer_SetPriority	192
tibemsMsgProducer_SetTimeToLive	193
tibemsQueueSender	194
tibemsQueueSender_GetQueue	195
tibemsQueueSender_Send	196
tibemsTopicPublisher	199
tibemsTopicPublisher_GetTopic	200
tibemsTopicPublisher_Publish	201
Chapter 7 Requestor	205
tibemsMsgRequestor	206
tibemsMsgRequestor_Close	207
tibemsMsgRequestor_Request	208
tibemsQueueRequestor_Create	209
tibemsTopicRequestor_Create	211
Chapter 8 Connection	213
tibemsConnection	214
tibemsConnection_Close	216
tibemsConnection_Create	218
tibemsConnection_CreateSession	220
tibemsConnection_GetActiveURL	221
tibemsConnection_GetClientId	222
tibemsConnection_GetExceptionListener	223
tibemsConnection_SetClientId	224
tibemsConnection_SetExceptionListener	225
tibemsConnection_Start	226
tibemsConnection_Stop	227
tibemsQueueConnection	229
tibemsQueueConnection_Create	230
tibemsQueueConnection_CreateQueueSession	232
tibemsTopicConnection	233
tibemsTopicConnection_Create	234
tibemsTopicConnection_CreateTopicSession	236
tibemsExceptionCallback	237
SSL	238
SSL Functions	238
SSL Constants	238
tibemsSSL_GetTrace	239
tibemsSSL_OpenSSLVersion	240
tibemsSSL_SetTrace	241

tibemsSSLParams	242
tibemsSSLParams_AddIssuerCert	244
tibemsSSLParams_AddTrustedCert	246
tibemsSSLParams_Create	248
tibemsSSLParams_Destroy	249
tibemsSSLParams_GetIdentity	250
tibemsSSLParams_GetPrivateKey	251
tibemsSSLParams_SetAuthOnly	252
tibemsSSLParams_SetCiphers	253
tibemsSSLParams_SetCRLPath	254
tibemsSSLParams_SetCRLUpdateInterval	255
tibemsSSLParams_SetExpectedHostName	256
tibemsSSLParams_SetHostNameVerifier	257
tibemsSSLParams_SetIdentity	258
tibemsSSLParams_SetPrivateKey	259
tibemsSSLParams_SetRandData	260
tibemsSSLParams_SetRenegotiateInterval	262
tibemsSSLParams_SetRenegotiateSize	263
tibemsSSLParams_SetVerifyHost	264
tibemsSSLHostNameVerifier	266
 Chapter 9 Connection Factory	 269
tibemsConnectionFactory	270
tibemsConnectionFactory_Create	273
tibemsConnectionFactory_CreateConnection	274
tibemsConnectionFactory_CreateConnectionSSL	275
tibemsConnectionFactory_Destroy	277
tibemsConnectionFactory_GetSSLProxyHost	278
tibemsConnectionFactory_GetSSLProxyPort	279
tibemsConnectionFactory_GetSSLProxyUser	280
tibemsConnectionFactory_GetSSLProxyPassword	281
tibemsConnectionFactory_GetType	282
tibemsConnectionFactory_SetClientID	283
tibemsConnectionFactory_SetConnectAttemptCount	284
tibemsConnectionFactory_SetConnectAttemptDelay	285
tibemsConnectionFactory_SetMetric	286
tibemsConnectionFactory_SetReconnectAttemptCount	287
tibemsConnectionFactory_SetReconnectAttemptDelay	288
tibemsConnectionFactory_SetServerURL	289
tibemsConnectionFactory_SetSSLParams	290
tibemsConnectionFactory_SetSSLProxy	291
tibemsConnectionFactory_SetSSLProxyAuth	293
tibemsConnectionFactory_SetType	294
tibemsConnectionFactoryType	295

tibemsFactoryLoadBalanceMetric	296
tibemsQueueConnectionFactory	297
tibemsQueueConnectionFactory_CreateConnection	298
tibemsQueueConnectionFactory_CreateConnectionSSL	299
tibemsTopicConnectionFactory	301
tibemsTopicConnectionFactory_CreateConnection	302
tibemsTopicConnectionFactory_CreateConnectionSSL	303
Chapter 10 Session	305
tibemsSession	306
tibemsSession_Close	309
tibemsSession_Commit	310
tibemsSession_CreateBrowser	311
tibemsSession_CreateBytesMessage	312
tibemsSession_CreateConsumer	313
tibemsSession_CreateDurableSubscriber	315
tibemsSession_CreateMapMessage	318
tibemsSession_CreateMessage	319
tibemsSession_CreateProducer	320
tibemsSession_CreateStreamMessage	321
tibemsSession_CreateTemporaryQueue	322
tibemsSession_CreateTemporaryTopic	323
tibemsSession_CreateTextMessage	324
tibemsSession_DeleteTemporaryQueue	325
tibemsSession_DeleteTemporaryTopic	326
tibemsSession_GetAcknowledgeMode	327
tibemsSession_GetTransacted	328
tibemsSession_Recover	329
tibemsSession_Rollback	331
tibemsSession_Unsubscribe	332
tibemsAcknowledgeMode	333
tibemsQueueSession	335
tibemsQueueSession_CreateBrowser	336
tibemsQueueSession_CreateReceiver	337
tibemsQueueSession_CreateSender	338
tibemsQueueSession_CreateTemporaryQueue	339
tibemsQueueSession_DeleteTemporaryQueue	340
tibemsTopicSession	341
tibemsTopicSession_CreateDurableSubscriber	342
tibemsTopicSession_CreatePublisher	344
tibemsTopicSession_CreateSubscriber	345
tibemsTopicSession_CreateTemporaryTopic	347
tibemsTopicSession_DeleteTemporaryTopic	348

tibemsTopicSession_Unsubscribe	349
Chapter 11 Queue Browser	351
tibemsQueueBrowser	352
tibemsQueueBrowser_Close	353
tibemsQueueBrowser_GetMsgSelector	354
tibemsQueueBrowser_GetNext	355
tibemsQueueBrowser_GetQueue	356
Chapter 12 Name Server Lookup	357
tibemsLookupContext	358
tibemsLookupContext_Create	359
tibemsLookupContext_CreateExternal	361
tibemsLookupContext_Destroy	362
tibemsLookupContext_Lookup	363
tibemsLookupParams	365
tibemsLookupParams_Create	366
tibemsLookupParams_Destroy	367
tibemsLookupParams_GetLdapServerUrl	368
tibemsLookupParams_SetLdapBaseDN	369
tibemsLookupParams_SetLdapCAFile	370
tibemsLookupParams_SetLdapCAPath	371
tibemsLookupParams_SetLdapCertFile	372
tibemsLookupParams_SetLdapCiphers	373
tibemsLookupParams_SetLdapConnType	374
tibemsLookupParams_SetLdapCredential	375
tibemsLookupParams_SetLdapKeyFile	376
tibemsLookupParams_SetLdapPrincipal	377
tibemsLookupParams_SetLdapRandFile	378
tibemsLookupParams_SetLdapSearchScope	379
tibemsLookupParams_SetLdapServerUrl	380
Chapter 13 XA—External Transaction Manager	381
tibemsXAConnection	382
tibemsXAConnection_Close	383
tibemsXAConnection_Create	385
tibemsXAConnection_CreateXASession	387
tibemsXAConnection_Get	388
tibemsXAConnection_GetXASession	389
XID	390
tibemsXAResource	391
tibemsXAResource_Commit	392

tibemsXAResource_End	393
tibemsXAResource_Forget	394
tibemsXAResource_GetRMID	395
tibemsXAResource_GetTransactionTimeout	396
tibemsXAResource_isSameRM	397
tibemsXAResource_Prepare	398
tibemsXAResource_Recover	399
tibemsXAResource_Rollback	400
tibemsXAResource_SetRMID	401
tibemsXAResource_SetTransactionTimeout	402
tibemsXAResource_Start	403
tibemsXASession	404
tibemsXASession_Close	405
tibemsXASession_GetSession	406
tibemsXASession_GetXAResource	407
Chapter 14 Types	409
Uniform Types	410
Chapter 15 Utilities	411
Utility Functions	412
tibems_Close	413
tibems_GetConnectAttemptCount	414
tibems_GetConnectAttemptDelay	415
tibems_GetReconnectAttemptCount	416
tibems_GetReconnectAttemptDelay	417
tibems_GetSocketReceiveBufferSize	418
tibems_GetSocketSendBufferSize	419
tibems_SetConnectAttemptCount	420
tibems_SetConnectAttemptDelay	421
tibems_SetReconnectAttemptCount	422
tibems_SetReconnectAttemptDelay	423
tibems_SetSocketReceiveBufferSize	424
tibems_SetSocketSendBufferSize	425
tibems_Sleep	426
tibems_Version	427
Chapter 16 Exception	429
tibems_status	430
tibemsStatus_GetText	440

Chapter 17 IBM Mainframe 441

IBM Mainframe Calls. 442

 tibems_SetCodePages() 443

 tibx_MVSConsole_SetConsumer() 445

 tibx_MVSConsole_Create() 446

Console_Response 448

Index 449

Tables

Table 1	Feature Support	3
Table 2	Linker Flags for 32-Bit UNIX	6
Table 3	Linker Flags for 64-Bit UNIX	6
Table 4	Dynamic Library Files for Microsoft Windows	8
Table 5	Static Library Files for Microsoft Windows	8
Table 6	Shareable Image Library Files for OpenVMS.	9
Table 7	Static Library Files for OpenVMS.	9
Table 8	JMS Message Headers	14
Table 9	Message Property Names	18
Table 10	Data Type Conversion	24
Table 11	Message Constants	27
Table 12	BytesMessage Read Functions	82
Table 13	BytesMessage Write Functions	89
Table 14	Message Field Type Indicators	135
Table 15	Destination Overview	140
Table 16	Certificate Encodings	238
Table 17	EMS Types.	410
Table 18	Status Codes	430

Preface

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 Rendezvous and TIBCO SmartSockets message products.



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.

Topics

- *Related Documentation, page xvi*
- *How to Contact TIBCO Customer Support, page xviii*

Related Documentation

This section lists documentation resources you may find useful.

TIBCO Enterprise Message Service Documentation

The following documents form the TIBCO Enterprise Message Service documentation set:

- *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 Installation* Read the relevant sections of this manual before installing this product.
- *TIBCO Enterprise Message Service Application Integration Guide* This manual presents detailed instructions for integrating TIBCO Enterprise Message Service with third-party products.
- *TIBCO Enterprise Message Service C & COBOL API Reference* This reference is available in HTML and PDF formats.
- *TIBCO Enterprise Message Service Java API Reference* This reference is available as JavaDoc, and you can access the reference only through the HTML documentation interface.
- *TIBCO Enterprise Message Service .NET API Reference* This reference is available in PDF and HTML format.
- *TIBCO Enterprise Message Service Release Notes* Release notes summarize new features, changes in functionality, and closed issues. 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 Rendezvous™ software
- TIBCO SmartSockets™ software

Third Party Documentation

- Java™ Message Service specification, available through java.sun.com/products/jms/index.html

- *Java™ Message Service* by Richard Monson-Haefel and David A. Chappell, O'Reilly and Associates, Sebastopol, California, 2001.

How to Contact TIBCO Customer Support

For comments or problems with this manual or the software it addresses, please contact TIBCO Support Services as follows.

- For an overview of TIBCO Support Services, and information about getting started with TIBCO Product Support, visit this site:

<http://www.tibco.com/services/support/default.jsp>

- If you already have a valid maintenance or support contract, visit this site:

<http://support.tibco.com>

Entry to this site requires a username and password. If you do not have a username, you can request one.

Chapter 1 **Introduction**

This chapter presents concepts specific to the TIBCO Enterprise Message Service™ C API and COBOL API. For more general information and concepts pertaining to TIBCO Enterprise Message Service (EMS) software, see the book *TIBCO Enterprise Message Service User's Guide*.

Topics

- *Overview, page 2*
- *Excluded Features and Restrictions, page 3*
- *Strings and Character Encodings, page 4*

Overview

TIBCO Enterprise Message Service C API implements (and extends) the JMS 1.1 specification. It is compatible with the JMS 1.0.2 specification.

TIBCO Enterprise Message Service C API closely mimics the Java API. This parallelism eases porting of programs between programming languages.

EMS COBOL API brings the power of EMS to OS/390 systems. Its entry points are 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.

Table 1 Feature Support

Feature	C	COBOL
XA protocols for external transaction managers	Yes	—
ConnectionConsumer, ServerSession, ServerSessionPool	—	—
Compression	Yes	—
SSL	Yes	—
Modify socket buffer sizes (see <code>tibems_SetSocketReceiveBufferSize</code> on page 424 and <code>tibems_SetSocketSendBufferSize</code> on page 425).	Yes	—

Strings and Character Encodings

C programs represent strings within messages as byte arrays. Before sending an outbound message, EMS programs translate strings to their byte representation using an encoding, which the program specifies. Conversely, when EMS programs receive inbound messages, they reconstruct strings from byte arrays using the same encoding.

When a program specifies an encoding, it applies to all strings in message bodies (names and values), and properties (names and values). It does *not* apply to header names nor values. The function `tibemsBytesMsg_WriteUTF` always uses UTF-8 as its encoding.

For a list of standard encoding names, see www.iana.org.

Outbound Messages	<p>C programs can determine the encoding of strings in outbound messages in two ways:</p> <ul style="list-style-type: none"> • Use the default global encoding—namely, UTF-8. • Set the encoding for an individual message using <code>tibemsMsg_SetEncoding</code> on page 68.
Inbound Messages	<p>An inbound message from another EMS client explicitly announces its encoding. A receiving client decodes the message using the proper encoding.</p>

IBM Mainframe—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()` on page 443.

Chapter 2 Programmer's Checklist

Developers of EMS programs can use this checklist during the five phases of the development cycle.

Install

- Install the EMS software release, which automatically includes the EMS client libraries, binaries, and header files in the `clients\c` subdirectory.

Code

- Application programs must add `clients\c\include` to the include path. (OpenVMS environments do not require an include path; skip this item.)
- Application programs must include the `tibems.h` header file:


```
#include <tibems/tibems.h>
```

Compile

- Compile programs with an ANSI-compliant C compiler.

Link

- Link with the appropriate EMS C library files; see [Link These Library Files](#) on page 6.

Run

- **UNIX** If you use dynamic EMS libraries on a UNIX platform, the environment variable `$LD_LIBRARY_PATH` must include the `clients/c/lib` directory (which contains the shared library file. (On some UNIX platforms, this variable is called `$LIBPATH`).
- **Windows** The `PATH` must include the `clients\c\bin` directory.
- **OpenVMS** The installation procedure automatically installs the shareable images required for using EMS dynamic libraries.
- **All Platforms** The application must be able to connect to a EMS server process (`tibemsd`).

Link These Library Files

EMS C programs must link the appropriate library files. Choose from the appropriate table based on operating system platform.

32-Bit UNIX

In 32-bit UNIX environments, both shared and static libraries are available. We recommend shared libraries to ease forward migration.

Table 2 Linker Flags for 32-Bit UNIX

Linker Flag	Description
-ltibems	All programs must link using this library flag.
-lssl	Programs that use SSL must link using these library flags.
-lcrypto	
-lz	Programs that use compression must link using this library flag.
-ltibemslookup	Programs that use EMS LDAP lookup must link using these library flags.
-ldap	
-llber	
-ltibjms	

64-Bit UNIX

In 64-bit UNIX environments, both shared and static libraries are available. We recommend shared libraries to ease forward migration. In this release, 64-bit libraries are available on HP-UX, Solaris, AIX and Linux (2.4 glibc 2.3) platforms.

To use 64-bit libraries, you must include *EMS_install_dir/clients/c/lib/64* in your library path, and it must precede any other EMS directory in the library path.

Table 3 Linker Flags for 64-Bit UNIX (Sheet 1 of 2)

Linker Flag	Description
-ltibems64	All programs must link using this library flag.

Table 3 Linker Flags for 64-Bit UNIX (Sheet 2 of 2)

Linker Flag	Description
-lssl64 -lcrypto64	Programs that use SSL must link using these library flags.
-lz64	Programs that use compression must link using this library flag.
-ltibemslookup64 -ldap64 -llber64 -ltibjms64	Programs that use EMS LDAP lookup must link using these library flags.

Microsoft Windows

For a list of Windows platforms that Release 4.3 supports, see the file `readme.txt` in the installation directory.

Both DLLs and static libraries are available. We recommend DLLs to ease forward migration.

Table 4 Dynamic Library Files for Microsoft Windows

Library File	Description
With dynamic libraries (DLLs), use the /MT compiler option.	
tibems.lib ws2_32.lib	All programs must link these libraries.
libtibemslookup.lib	Programs that use EMS LDAP lookup must link this library.
liboldap32.lib olber32.lib	In addition, programs that use EMS lookup must link one of these pairs of libraries.
libldap32_d.lib liblber32_d.lib	

Table 5 Static Library Files for Microsoft Windows

Library File	Description
With static libraries (DLLs), use the /MD compiler option.	
libtibems.lib ws2_32.lib ssleay32mt.lib libeay32mt.lib zlib.lib	All programs must link these libraries.
libtibemslookup.lib	Programs that use EMS LDAP lookup must link this library.
liboldap32.lib olber32.lib	In addition, programs that use EMS lookup must link one of these pairs of libraries.
libldap32_d.lib liblber32_d.lib	

OpenVMS

In OpenVMS environments, both shared and static libraries are available. We recommend shared libraries to ease forward migration.

Table 6 Shareable Image Library Files for OpenVMS

Library File	Description
LIBTIBEMSSHR.EXE	All programs must link this library.
LIBTIBEMSLOOKUPSHR.EXE	Programs that use EMS LDAP lookup must link these libraries.
LIBLDAPSHR.EXE	
LIBLBERSHR.EXE	
LIBXMLSHR.EXE	
LIBCRYPTOSHR.EXE	Programs that use SSL must link these libraries.
LIBSSLSHR.EXE	
LIBZSHR.EXE	Programs that use data compression must link this library.

Table 7 Static Library Files for OpenVMS

Library File	Description
LIBTIBEMS.OLB	All programs must link this library.
LIBTIBEMSLOOKUP.OLB	Programs that use EMS LDAP lookup must link these libraries.
LIBLDAP.OLB	
LIBLBER.OLB	
LIBXML.OLB	
LIBCRYPTO.OLB	Programs that use SSL must link these libraries.
LIBSSL.OLB	
LIBZ.OLB	Programs that use data compression must link this library.

Chapter 3 **Messages**

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

Topics

- *Parts of a Message, page 12*
- *Body Types, page 13*
- *Headers, page 14*
- *Properties, page 18*
- *Message Selectors, page 21*
- *tibemsMsg, page 25*
- *tibemsBytesMsg, page 76*
- *tibemsMapMsg, page 91*
- *tibemsObjectMsg, page 106*
- *tibemsStreamMsg, page 110*
- *tibemsTextMsg, page 125*

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 on page 13.
- **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` on page 138
 `tibemsMsg_GetBodyType` on page 40

Headers

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 Table 8.

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.

Table 8 JMS Message Headers (Sheet 1 of 4)

Description
Correlation ID 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: <ul style="list-style-type: none">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 <i>not</i> 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. tibemsMsg_GetCorrelationID on page 42 tibemsMsg_SetCorrelationID on page 64

Table 8 JMS Message Headers (Sheet 2 of 4)

Description
<p>Delivery Mode</p> <p>Delivery Mode instructs the server concerning persistent storage for the message.</p> <p>Sending calls record 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 <code>tibemsMsgProducer_SetDeliveryMode</code> on page 189.</p> <p><code>tibemsMsg_GetDeliveryMode</code> on page 44</p> <p><code>tibemsMsg_SetDeliveryMode</code> on page 66</p> <p>For values, see <code>tibemsDeliveryMode</code> on page 130.</p>
<p>Destination</p> <p>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.</p> <p>Listeners that consume messages from several destinations can use this property to determine the actual destination of a message.</p> <p><code>tibemsMsg_GetDestination</code> on page 45</p> <p><code>tibemsMsg_SetDestination</code> on page 67</p>
<p>Expiration</p> <p>Sending calls record the expiration time (in milliseconds) of the message in this field:</p> <ul style="list-style-type: none"> • 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. <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><code>tibemsMsg_GetExpiration</code> on page 47</p> <p><code>tibemsMsg_SetExpiration</code> on page 69</p>

Table 8 JMS Message Headers (Sheet 3 of 4)

Description
Message ID 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 <code>tibemsMsgProducer_SetDisableMessageID</code> on page 190. When the producer disables IDs, the value of this header is null. <code>tibemsMsg_GetMessageID</code> on page 48 <code>tibemsMsg_SetMessageID</code> on page 70
Priority Sending calls record the priority of a message in this header, based on either a property of the producer (<code>tibemsMsgProducer_SetPriority</code> on page 192), 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.) <code>tibemsMsg_GetPriority</code> on page 49 <code>tibemsMsg_SetPriority</code> on page 71
Redelivered The server sets this header to indicate whether a message might duplicate a previously delivered message: <ul style="list-style-type: none"><code>false</code>—The server has <i>not</i> previously attempted to deliver this message to the consumer.<code>true</code>—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 acknowledgement. <code>tibemsMsg_GetRedelivered</code> on page 51 <code>tibemsMsg_SetRedelivered</code> on page 72 See also, <code>tibemsAcknowledgeMode</code> on page 333

Table 8 JMS Message Headers (Sheet 4 of 4)

Description
Reply To <p>Sending clients can set this header to request that recipients reply to the message:</p> <ul style="list-style-type: none"> When the value is a destination object, recipients can send replies to that destination. Such a message is called a <i>request</i>. When the value is null, the sender does not expect a reply. <p>When sending a reply, clients can refer to the corresponding request by setting the Correlation ID.</p> <p><code>tibemsMsg_GetReplyTo</code> on page 52</p> <p><code>tibemsMsg_SetReplyTo</code> on page 73</p>
Timestamp <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 <code>tibemsMsgProducer_SetDisableMessageTimestamp</code> on page 191. When the producer disables timestamps, the value of this header is zero.</p> <p><code>tibemsMsg_GetTimestamp</code> on page 53</p> <p><code>tibemsMsg_SetTimestamp</code> on page 74</p>
Type <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><code>tibemsMsg_GetType</code> on page 54</p> <p><code>tibemsMsg_SetType</code> on page 75</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 on page 21.

Setting Message Properties

Property names must conform to the syntax for message selector identifiers; see Identifiers on page 21.

Property values must *not* be null, nor the 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.

Table 9 Message Property Names (Sheet 1 of 2)

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

Table 9 Message Property Names (Sheet 2 of 2)

Property	Description
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 JMS_TIBCO_SENDER.
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 <code>true</code> .
JMS_TIBCO_MSG_EXT	Producers can set this property to <code>true</code> to indicate that a message uses EMS extensions to the JMS specification for messages. The server sets this property to <code>true</code> 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"> When the property value is <code>null</code>, the trace output contains the message ID and sequence number. When the property value is <code>body</code>, the trace output includes the message body as well.
JMS_TIBCO_PRESERVE_UNDELIVERED	When this property is <code>true</code> , 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 JMS_TIBCO_DISABLE_SENDER.
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 <code>_</code> (underscore) and <code>\$</code> (dollar).
Illegal	Certain names are exceptions, which cannot be used as identifiers. In particular, <code>NULL</code> , <code>TRUE</code> , <code>FALSE</code> , <code>NOT</code> , <code>AND</code> , <code>OR</code> , <code>BETWEEN</code> , <code>LIKE</code> , <code>IN</code> , <code>IS</code> , and <code>ESCAPE</code> 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 <code>NULL</code> .

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, <code>'literal''s'</code> . 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 <code>57</code> , <code>-957</code> , and <code>+62</code> ; numbers in the range of <code>long</code> are supported.
Approximate Numeric Literal	An approximate numeric literal is a numeric value with a decimal point (such as <code>7.</code> , <code>-95.7</code> , and <code>+6.2</code>), or a numeric value in scientific notation (such as <code>7E3</code> and <code>-57.9E2</code>); numbers in the range of <code>double</code> are supported. Approximate literals use the floating-point literal syntax of the Java programming language.
Boolean Literal	The boolean literals are <code>TRUE</code> and <code>FALSE</code> (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 <code>true</code> matches the message; a selector that evaluates to <code>false</code> or <code>unknown</code> 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: <code>NOT</code> , <code>AND</code> , <code>OR</code> .
Comparison Operators	<p>Comparison operators: <code>=</code>, <code>></code>, <code>>=</code>, <code><</code>, <code><=</code>, <code><></code> (not equal).</p> <p>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 <code>false</code>. If either value in a comparison evaluates to <code>NULL</code>, then the result is <code>unknown</code> (in SQL 3-valued logic).</p> <p>Comparison of string values is restricted to <code>=</code> and <code><></code>. Two strings are equal if and only if they contain the same sequence of characters.</p> <p>Comparison of boolean values is restricted to <code>=</code> and <code><></code>.</p>
Arithmetic Operators	<p>Arithmetic operators in precedence order:</p> <ul style="list-style-type: none"> • <code>+</code>, <code>-</code> (unary) • <code>*</code>, <code>/</code> (multiplication and division) • <code>+</code>, <code>-</code> (addition and subtraction) <p>Arithmetic operations obey numeric promotion rules of the Java programming language.</p>

Between Operator	<p><i>arithmetic-expr1</i> [NOT] BETWEEN <i>arithmetic-expr2</i> AND <i>arithmetic-expr3</i></p> <p>The BETWEEN comparison operator includes its endpoints. For example:</p> <ul style="list-style-type: none"> • <code>age BETWEEN 5 AND 9</code> is equivalent to <code>age >= 5 AND age <= 9</code> • <code>age NOT BETWEEN 5 AND 9</code> is equivalent to <code>age < 5 OR age > 9</code>
String Set Membership	<p><i>identifier</i> [NOT] IN (<i>string-literal1</i>, <i>string-literal2</i>, ...)</p> <p>The <i>identifier</i> must evaluate to either a string or NULL. If it is NULL, then the value of this expression is unknown.</p>
Pattern Matching	<p><i>identifier</i> [NOT] LIKE <i>pattern-value</i> [ESCAPE <i>escape-character</i>]</p> <p>The <i>identifier</i> must evaluate to a string.</p> <p>The <i>pattern-value</i> is a string literal, in which some characters bear special meaning:</p> <ul style="list-style-type: none"> • <code>_</code> (underscore) can match any single character. • <code>%</code> (percent) can match any sequence of zero or more characters. • <i>escape-character</i> preceding either of the special characters changes them into ordinary characters (which match only themselves).
Null Header or Property	<p><i>identifier</i> IS NULL</p> <p>This comparison operator tests whether a message header is null, or a message property is absent.</p> <p><i>identifier</i> IS NOT NULL</p> <p>This comparison operator tests whether a message header or message property is non-null.</p>

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

Table 10 summarizes legal datatype conversions. The symbol X in Table 10 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.

Table 10 Data Type Conversion

	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	
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 carry information among EMS client programs.

Related Types tibemsBytesMsg, tibemsMapMsg, tibemsObjectMsg, tibemsStreamMsg, tibemsTextMsg

(Sheet 1 of 3)

Function	Description	Page
Receipt		
tibemsMsg_Acknowledge	Acknowledge messages.	28
Body		
tibemsMsg_GetBodyType	Get the body type of a message.	40
Message Life Cycle		
tibemsMsg_ClearBody	Clear the body of a message.	30
tibemsMsg_ClearProperties	Clear the properties of a message.	31
tibemsMsg_Create	Create a message object.	32
tibemsMsg_CreateCopy	Create a copy of the message object.	33
tibemsMsg_CreateFromBytes	Create a message object from data in a byte sequence.	34
tibemsMsg_Destroy	Destroy a message.	35
tibemsMsg_MakeWriteable	Make a message writeable.	55
Output		
tibemsMsg_GetAsBytes	Format message data into a byte sequence.	36
tibemsMsg_GetAsBytesCopy	Format message data as a byte sequence into storage supplied by the program.	38
tibemsMsg_GetByteSize	Compute the size of the message as a byte sequence in EMS wire format.	41

(Sheet 2 of 3)

Function	Description	Page
<code>tibemsMsg_Print</code>	Print a message.	56
<code>tibemsMsg_PrintFile</code>		
Headers and Properties		
For details, see Headers on page 14.		
<code>tibemsMsg_ClearProperties</code>	Clear the properties of a message.	31
<code>tibemsMsg_GetCorrelationID</code>	Get the correlation ID header of a message.	42
<code>tibemsMsg_GetDeliveryMode</code>	Get the delivery mode header from a message.	44
<code>tibemsMsg_GetDestination</code>	Get the destination header from a message.	45
<code>tibemsMsg_GetEncoding</code>	Get the character encoding header from a message.	46
<code>tibemsMsg_GetExpiration</code>	Get the expiration header from a message.	47
<code>tibemsMsg_GetMessageID</code>	Get the message ID header from a message.	48
<code>tibemsMsg_GetPriority</code>	Get the priority header from a message.	49
<code>tibemsMsg—Properties Get</code>	Get the value of a message property.	57
<code>tibemsMsg_GetPropertyNames</code>	Get a list of property names from a message.	50
<code>tibemsMsg_GetRedelivered</code>	Get the redelivered header from a message.	51
<code>tibemsMsg_GetReplyTo</code>	Get the reply-to header from a message.	52
<code>tibemsMsg_GetTimestamp</code>	Get the timestamp header from a message.	53
<code>tibemsMsg_GetType</code>	Get the type header of a message.	54
<code>tibemsMsg_PropertyExists</code>	Test whether a named property has been set on a message.	63
<code>tibemsMsg_SetCorrelationID</code>	Set the correlation ID header of a message.	64
<code>tibemsMsg_SetDeliveryMode</code>	Set the delivery mode header of a message.	66
<code>tibemsMsg_SetDestination</code>	Set the destination header of a message.	67

(Sheet 3 of 3)

Function	Description	Page
<code>tibemsMsg_SetEncoding</code>	Set the character encoding header of a message.	68
<code>tibemsMsg_SetExpiration</code>	Set the expiration header of a message.	69
<code>tibemsMsg_SetMessageID</code>	Set the message ID header of a message.	70
<code>tibemsMsg_SetPriority</code>	Set the priority header of a message.	71
<code>tibemsMsg_SetRedelivered</code>	Set the redelivered header of a message.	72
<code>tibemsMsg_SetReplyTo</code>	Set the reply-to header of a message.	73
<code>tibemsMsg_SetTimestamp</code>	Set the timestamp header of a message.	74
<code>tibemsMsg_SetType</code>	Set the type header of a message.	75
<code>tibemsMsg—Properties Set</code>	Set the value of a message property.	60

Table 11 Message Constants

Constant	Description
<code>TIBEMS_DEFAULT_DELIVERY_MODE</code>	<p><code>TIBEMS_PERSISTENT</code></p> <p>When neither the sending call nor the producer supplies a delivery mode, this default applies.</p>
<code>TIBEMS_DEFAULT_PRIORITY</code>	<p>4</p> <p>When neither the sending call nor the producer supplies a priority, this default applies.</p> <p>See also, Priority on page 16.</p>
<code>TIBEMS_DEFAULT_TIME_TO_LIVE</code>	<p>0</p> <p>When neither the sending call nor the producer supplies a priority, this default applies. The default value, zero, indicates that messages do not expire.</p> <p>See also Expiration on page 15.</p>

tibemsMsg_Acknowledge

Function

Purpose


C Declaration

COBOL Call

Acknowledge messages.

tibems_status tibemsMsg_Acknowledge(
 tibemsMsg message);

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 acknowledgement 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 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.)
- In all other modes, this call has no effect. In particular, modes that specify transactions or implicit acknowledgement 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` on page 166
 `tibemsSession` on page 306
 `tibemsAcknowledgeMode` on page 333

tibemsMsg_ClearBody

Function

Purpose

C Declaration

COBOL Call

Clear the body of a message.

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

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



message has usage pointer.

Parameters

Parameter	Description
message	Clear this message.

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	Clear this message.

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	The function stores a pointer to the new message in this location.

Remarks This call creates a new message.

This function allocates the storage for the 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` on page 35

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	The function stores a pointer to the new copy in this location.

Remarks

This call creates 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` on page 35

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	The function stores a pointer to the new message in this location.
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 <code>tibemsMsg_GetAsBytes</code> or <code>tibemsMsg_GetAsBytesCopy</code> .

Remarks This call creates 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` on page 35
`tibemsMsg_GetAsBytes` on page 36
`tibemsMsg_GetAsBytesCopy` on page 38
`tibemsMsg_MakeWriteable` on page 55

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 Format message data into a byte sequence.

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.

Parameter	Description
message	Fill the byte array with the content of this message.
bytes	The function allocates a byte sequence, and stores a pointer to it in this location.
actual_size	The function stores the length of the byte sequence in 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` on page 34

`tibemsMsg_GetAsBytesCopy` on page 38

`tibemsMsg_GetByteSize` on page 41

tibemsMsg_GetAsBytesCopy

Function

Purpose Format message data as a byte sequence 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.

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

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
TIBEMS_INSUFFICIENT_BUFFER	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](#) on page 34
 [tibemsMsg_GetAsBytes](#) on page 36
 [tibemsMsg_GetByteSize](#) on page 41

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` on page 54.

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
<code>message</code>	Get the body type of this message.
<code>type</code>	The function stores the body type in this location.

See Also Body Types on page 13
`tibemsMsgType` on page 138

tibemsMsg_GetByteSize

Function

Purpose Compute the size of the message as a byte sequence in EMS wire format.

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.

Parameter	Description
message	Compute the size of this message.
size	The function stores the message size in this location.

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` on page 38

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	The function stores the correlation ID in this location.

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 on page 48
 tibemsMsg_SetCorrelationID on page 64

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	The function stores the delivery mode in this location.

Remarks Delivery mode is a header property of message objects.

See Also tibemsDeliveryMode on page 130
tibemsMsg_SetDeliveryMode on page 66

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	The function stores the destination in this location.

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 on page 144](#)
[tibemsMsg_SetDestination on page 67](#)

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	The function stores the character encoding in this location.

- 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. The functions `tibemsBytesMsg_ReadUTF` and `tibemsBytesMsg_WriteUTF` are exempt from message encoding settings.
- See Also** Strings and Character Encodings on page 4
`tibemsMsg_SetEncoding` on page 68

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	The function stores the expiration in this location.

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 on page 69
tibemsMsgProducer_GetTimeToLive on page 185

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	The function stores the message ID in this location.

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 on page 190. When the producer disables IDs, the value of this header is null.

See Also tibemsMsg_GetCorrelationID on page 42
tibemsMsg_SetMessageID on page 70
tibemsMsgProducer_SetDisableMessageID on page 190

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	The function stores the priority in this location.

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 on page 71
tibemsMsgProducer_Send on page 186
tibemsMsgProducer_SetPriority on page 192

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	The function stores the property names in this location.

See Also tibemsMsgEnum on page 131

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	The function stores 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 the server has previously attempted to deliver this message to the consumer, but the consumer did not return timely acknowledgement.

See Also tibemsMsg_SetRedelivered on page 72

tibemsMsg_GetReplyTo

Function

Purpose

C Declaration

COBOL Call

Get the reply-to header from a message.

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

```
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	The function stores the reply-to header in this location.

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 on page 64

tibemsMsg_SetReplyTo on page 73

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	The function stores the timestamp in this location.

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` on page 191. When the producer disables timestamps, the value of this header is zero.

See Also `tibemsMsg_SetTimestamp` on page 74
`tibemsMsgProducer_SetDisableMessageTimestamp` on page 191

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` on page 40.

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	The function stores the type in this location.

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` on page 75

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 );  
  
void tibemsMsg_PrintFile(  
    tibemsMsg message,  
    FILE* file );
```
- COBOL Call

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



message has usage pointer.
COBOL does not support tibemsMsg_PrintFile in release 4.3.

Parameters	Parameter	Description
	message	Print this message.
	file	Output to this file.

- Remarks

These calls print 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.

tibemsMsg_PrintFile prints the message to the file that you specify.

tibemsMsg—Properties Get

Function

Purpose Get the value of a message property.

C Declaration

```

tibems_status tibemsMsg_GetProperty(
    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_GetProperty"
  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 Message Selectors on page 21). Property names must not be null, and must not be empty strings.
value	The call stores the value in this location.

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
TIBEMS_CONVERSION_FAILED	The actual type of the property is not compatible with the requested type.
TIBEMS_NOT_FOUND	The property is not set in this message.

See Also tibemsMsg—Properties Set on page 60

tibemsMsg—Properties Set

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 Message Selectors on page 21). 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 Message Selectors on page 21). 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
TIBEMS_MSG_NOT_WRITEABLE	The message is read-only.

See Also tibemsMsg—Properties Get on page 57

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	The function stores the boolean result of the test in this location: <ul style="list-style-type: none">TIBEMS_TRUE if the property has a value on the messageTIBEMS_FALSE otherwise

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 on page 42](#)
 [tibemsMsg_GetMessageID on page 48](#)

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 on page 130](#)
[tibemsMsg_GetDeliveryMode on page 44](#)

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 on page 144
- tibemsMsg_GetDestination on page 45
- tibemsQueueSender_GetQueue on page 195
- tibemsTopicPublisher_GetTopic on page 200

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	Set the character encoding of this message.
value	Set the character encoding to this value. For a list of standard encoding names, see www.iana.org .

Remarks This call extends the JMS specification.

Programs can set the encoding for individual messages.

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. The functions `tibemsBytesMsg_ReadUTF` and `tibemsBytesMsg_WriteUTF` are exempt from message encoding settings.

See Also Strings and Character Encodings on page 4
`tibemsMsg_GetEncoding` on page 46

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](#) on page 47
[tibemsMsgProducer_GetTimeToLive](#) on page 185
[tibemsMsgProducer_SetTimeToLive](#) on page 193

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` on page 190. When the producer disables IDs, the value of this header is null.

See Also `tibemsMsg_GetCorrelationID` on page 42
`tibemsMsg_GetMessageID` on page 48
`tibemsMsg_SetCorrelationID` on page 64
`tibemsMsgProducer_SetDisableMessageID` on page 190

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.
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](#) on page 49
[tibemsMsgProducer_GetPriority](#) on page 184
[tibemsMsgProducer_SetPriority](#) on page 192

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 acknowledgement.

See Also tibemsMsg_GetRedelivered on page 51

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](#) on page 52
[tibemsMsg_SetCorrelationID](#) on page 64

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` on page 191. When the producer disables timestamps, the value of this header is zero.

See Also `tibemsMsg_GetTimestamp` on page 53
`tibemsMsgProducer_SetDisableMessageTimestamp` on page 191

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` on page 40.

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` on page 54

tibemsBytesMsg

Type

- Purpose** A message containing a stream of uninterpreted bytes.
- Related Types** `tibemsMsg` on page 25
- Remarks** Messages with this body type contain a single value, which is a byte sequence.

Function	Description	Page
<code>tibemsBytesMsg_Create</code>	Create a bytes message.	77
<code>tibemsBytesMsg_GetBodyLength</code>	Get the body length (in bytes) of a bytes message.	78
<code>tibemsBytesMsg_GetBytes</code>	Get the body data of a bytes message.	79
<code>tibemsBytesMsg—Read</code>	Read primitive datatypes from the byte stream in the message body.	80
<code>tibemsBytesMsg_ReadBytes</code>	Read bytes to a byte sequence from the byte stream in the message body.	84
<code>tibemsBytesMsg_Reset</code>	Set the read position to the beginning of the byte stream, and mark the message body as read-only.	85
<code>tibemsBytesMsg_SetBytes</code>	Set the body data of a bytes message from a byte sequence.	86
<code>tibemsBytesMsg—Write</code>	Write primitive datatypes to the byte stream in the message body.	87
<code>tibemsBytesMsg_WriteBytes</code>	Write bytes from a byte array to the byte stream in the message body.	90

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	The function stores a pointer to the new message in this location.

Remarks This call creates a new bytes message.

This function allocates the storage for the 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_Create` on page 32
`tibemsMsg_Destroy` on page 35

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	The function stores the body length in this location.

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.

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

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 );

tibems_status tibemsBytesMsg_ReadUTF(
    tibemsBytesMsg message,
    const char** value,
    tibems_int* length );

```

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.

CALL "tibemsBytesMsg_ReadUTF"

```

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



message has usage pointer in all calls.
value has usage pointer in tibemsBytesMsg_ReadUTF.

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.

Table 12 BytesMessage Read Functions

Function	# Bytes	Interpret As
tibemsBytesMsg_ReadBoolean	1	tibems_bool
tibemsBytesMsg_ReadByte	1	tibems_byte
tibemsBytesMsg_ReadUnsignedByte	1	tibems_int
tibemsBytesMsg_ReadShort	2	tibems_short
tibemsBytesMsg_ReadUnsignedShort	2	tibems_int
tibemsBytesMsg_ReadChar	2	tibems_wchar
tibemsBytesMsg_ReadInt	4	tibems_int
tibemsBytesMsg_ReadLong	8	tibems_long
tibemsBytesMsg_ReadFloat	4	tibems_float
tibemsBytesMsg_ReadDouble	8	tibems_double
tibemsBytesMsg_ReadUTF	varies	char* Encoded as UTF.

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

Parameter	Description
value	The function stores the datum in this location.
length	<code>tibemsBytesMsg_ReadUTF</code> reads a UTF-8 string. Since the length of the string cannot be determined in advance, the function stores the actual length in this location.

See Also `tibemsBytesMsg_ReadBytes` on page 84

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.

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

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` on page 30, 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.

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 );

tibems_status tibemsBytesMsg_WriteUTF(
    tibemsBytesMsg message,
    const char* 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.  
  
CALL "tibemsBytesMsg_WriteUTF"  
    USING BY VALUE message,  
        BY REFERENCE value,  
        RETURNING tibems-status  
END-CALL.
```



message has usage pointer in all calls.

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.

Table 13 BytesMessage Write Functions

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	
<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	
<code>tibemsBytesMsg_WriteUTF</code>	varies	For information about the UTF-8 format, see <i>File System Safe UCS Transformation Format (FSS_UFT)</i> , X/Open Preliminary Specification, X/Open Company Ltd., Document Number: P316. This information also appears in ISO/IEC 10646, Annex P.

See Also `tibemsBytesMsg_WriteBytes` on page 90

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 on page 25
- Remarks** Messages with this body type contain several values, indexed by name.

Function	Description	Page
tibemsMapMsg_Create	Create a map message.	92
tibemsMapMsg—Get	Get an enumeration of the field names in a map message.	93
tibemsMapMsg_GetMapNames	Create a map message.	97
tibemsMapMsg_ItemExists	Test that a named pair is set.	98
tibemsMapMsg—Set	Set a name-value pair in a map message.	99

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 on page 87 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	The function stores a pointer to the new message in this location.

Remarks

This call creates a new map message.

This function allocates the storage for the 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_Create` on page 32

`tibemsMsg_Destroy` on page 35

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	<code>tibemsMapMsg_GetBytes</code> stores a pointer to the byte sequence in this location.
bytesSize	<code>tibemsMapMsg_GetBytes</code> 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` on page 134.

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	The function stores a pointer to the enumeration in this location.

See Also tibemsMsgEnum on page 131

tibemsMapMsg_ItemExists

Function

- Purpose

Test that a named pair is set.
- 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 in this location.

tibemsMapMsg—Set

Function

Purpose	Set a name-value pair in a map message.
Single Value C Declarations	<pre> 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); </pre>
Array C Declarations	<pre> tibems_status tibemsMapMsg_SetDoubleArray(tibemsMapMsg message, const char* name, const tibems_double* value, </pre>

```

        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.

Parameter	Description
message	Set the pair in the body of this map message.

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

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` on page 93
`tibemsMapMsg_SetBytes` on page 104

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 on page 25

Function	Description	Page
tibemsObjectMsg_Create	Create an object message.	107
tibemsObjectMsg_GetObjectBytes	Get the byte sequence representing a serialized object from a message.	108
tibemsObjectMsg_SetObjectBytes	Set the byte sequence of an object message.	109

Object Messages in C 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	The function stores a pointer to the new message in this location.

Remarks This call creates a new object message.

This function allocates the storage for the 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_Create` on page 32
`tibemsMsg_Destroy` on page 35

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	The function stores a pointer to the byte sequence (within the message) in this location.
byteSize	The function stores the length of the byte sequence in this location.

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.
byteSize	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 on page 25

Member	Description	Page
tibemsStreamMsg_Create	Create a stream message.	112
tibemsStreamMsg_FreeField	Free allocated storage hidden in a field read from a stream message.	113
tibemsStreamMsg—Read	Read primitive datatypes from a stream message.	114
tibemsStreamMsg_ReadBytes	Read a byte array from a stream message.	117
tibemsStreamMsg_ReadField	Read a field from a stream message.	118
tibemsStreamMsg_Reset	Set the read position to the beginning of the stream, and mark the message body as read-only.	119
tibemsStreamMsg—Write	Write data to a stream message.	120
tibemsBytesMsg_WriteBytes	Write bytes from a byte array to a stream message.	124

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 on page 87 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	The function stores a pointer to the new message in this location.

Remarks This call creates a new stream message.

This function allocates the storage for the 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_Create` on page 32
`tibemsMsg_Destroy` on page 35

tibemsStreamMsg_FreeField

Function

Purpose Free allocated storage hidden in a field read from 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` creates a field struct. Field structs can contain data in allocated storage. Programs that call `tibemsStreamMsg_ReadField` *must* subsequently call this function to free that allocated storage.

This function does not apply after calls to `tibemsMapMsg_GetField`.

See Also `tibemsMsgField` on page 134
`tibemsStreamMsg_ReadField` on page 118

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	The function stores a pointer to the field struct in this location.

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` on page 117
`tibemsStreamMsg_ReadField` on page 118

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	The function stores a pointer to the byte sequence (within the message) in this location.
length	The function stores the actual number of bytes read in this location.

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 on page 114
tibemsStreamMsg_ReadField on page 118

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 in this location.

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` on page 113

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` on page 30, 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	<pre> 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); </pre>
Array C Declarations	<pre> 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); </pre>

```

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).

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 on page 124

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 on page 25
- Remarks** Messages with this body type contain a single value, which is a string.

Function	Description	Page
tibemsTextMsg_Create	Create a text message.	126
tibemsTextMsg_GetText	Get the string data from a text message.	127
tibemsTextMsg_SetText	Set the data string of a text message.	128

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	The function stores a pointer to the new message in this location.

Remarks

This call creates a new text message.

This function allocates the storage for the 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_Create` on page 32

`tibemsMsg_Destroy` on page 35

See Also

`tibemsSession_CreateTextMessage` on page 324

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	The function stores a pointer to the string (within the message) in this location.

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

See Also `tibemsMsgField` on page 134

tibemsDeliveryMode

Type

Purpose	Define delivery mode constants.		
C Declaration	<pre>typedef enum { TIBEMS_NON_PERSISTENT, TIBEMS_PERSISTENT, TIBEMS_RELIABLE } tibemsDeliveryMode;</pre>		
COBOL	<pre>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.</pre>		

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 Reliable Message Delivery on page 70 in <i>TIBCO Enterprise Message Service User's Guide</i> .

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	Page
<code>tibemsMsgEnum_Destroy</code>	Destroy a message enumerator.	132
<code>tibemsMsgEnum_GetNextName</code>	Get the next item from a message enumerator.	133

- See Also**`tibemsMsg_GetPropertyNames` on page 50
`tibemsMapMsg_GetMapNames` on page 97

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	The function stores the next item in this location.

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.

(Sheet 1 of 2)

Field	Description
type	A one-byte indicator of the field’s datatype; for values, see Table 14 below.
size	The size of the data (in bytes). Zero is a special value, indicating that the size is unknown.

(Sheet 2 of 2)

Field	Description
count	If the data is an array, this value is the number of elements in the array.
data	The actual data in the field, or the property value.

Table 14 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	
TIBEMS_LONG_ARRAY	22	
TIBEMS_FLOAT_ARRAY	23	
TIBEMS_DOUBLE_ARRAY	24	

See Also `tibemsMsg_GetProperty`, listed at `tibemsMsg—Properties Get` on page 57
 `tibemsMapMsg_GetField`, listed at `tibemsMapMsg—Get` on page 93
 `tibemsStreamMsg_FreeField` on page 113
 `tibemsStreamMsg_ReadField` on page 118
 `tibemsMsgEnum_GetNextName` on page 133

tibemsMsgField_Print

Function

Purpose Print a message field.

C Declaration

```
void tibemsMsgField_Print(  
    tibemsMsgField* field );  
  
void tibemsMsgField_PrintFile(  
    tibemsMsgField* field,  
    FILE* file);
```

COBOL Call

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



COBOL does not support tibemsMsgField_PrintFile in release 4.3.

Remarks The first form prints to stdout. The second form prints to a file.
Both forms print in the format *FieldDataType: Value*.

Parameters	Parameter	Description
	field	Print this message field instance.
	file	Output to this file.

tibemsMsgType

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 on page 40

Chapter 4 **Destination**

Topics

- *Destination Overview, page 140*
- *tibemsDestination, page 144*
- *tibemsQueue, page 150*
- *tibemsTemporaryQueue, page 154*
- *tibemsTemporaryTopic, page 155*
- *tibemsTopic, page 156*

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.

Table 15 Destination Overview (Sheet 1 of 3)

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.

Table 15 Destination Overview (Sheet 2 of 3)

Aspect	Static	Dynamic	Temporary
Creation	Administrators create static destinations using EMS server administration tools or API.	<p>If the server configuration permits dynamic destinations, client programs can create one in two steps:</p> <ol style="list-style-type: none"> 1. Create a local destination object; see <code>tibemsSession</code> on page 306. 2. Send a message to that destination, or create a consumer for it. Either of these actions automatically creates the destination in the server. 	Client programs create temporary destinations; see <code>tibemsSession</code> on page 306.
Lookup	Client programs lookup static destinations by name. Successful lookup returns a local object representation of the destination; see <code>tibemsLookupContext_Lookup</code> on page 363.	Client programs lookup dynamic destinations by name. Successful lookup returns a local object representation of the destination; see <code>tibemsLookupContext_Lookup</code> on page 363.	Not applicable.

Table 15 Destination Overview (Sheet 3 of 3)

Aspect	Static	Dynamic	Temporary
Duration	A static destination remains in the server until an administrator explicitly deletes it.	<div>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:<ul style="list-style-type: none">• Topic or Queue all client programs that access the destination have disconnected• Topic no offline durable subscribers exist for the topic• Queue queue, no messages are stored in the queue</div>	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](#) on page 146
[tibemsDestination_GetType](#) on page 149

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	Page
<code>tibemsDestination_Copy</code>	Create an independent copy of a destination object.	145
<code>tibemsDestination_Create</code>	Create a destination object.	146
<code>tibemsDestination_Destroy</code>	Destroy a destination object.	147
<code>tibemsDestination_GetName</code>	Get the name of a destination object.	148
<code>tibemsDestination_GetType</code>	Get the type of a destination object.	149

Related Types

`tibemsQueue` on page 150
`tibemsTemporaryQueue` on page 154
`tibemsTopic` on page 156
`tibemsTemporaryTopic` on page 155

See Also

`tibemsMsg_GetDestination` on page 45
`tibemsMsg_SetDestination` on page 67
`tibemsLookupContext` on page 358

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	The function stores the new copy in this location.

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	The function stores 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 on page 143

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.

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.

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 on page 143

tibemsQueue

Type

Purpose Queues deliver each message to exactly one consumer.

Function	Description	Page
tibemsQueue_Create	Create a queue object.	151
tibemsQueue_Destroy	Destroy a queue object.	152
tibemsQueue_GetQueueName	Get the name of a queue object.	153

Related Types tibemsDestination on page 144
 tibemsTemporaryQueue on page 154

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.

Remarks This call creates only local objects (within the program). It does not attempt to lookup the corresponding server object until the program creates a `tibemsMsgConsumer` or a `tibemsMsgProducer` that uses the queue. That automatic lookup can result in either of two outcomes:

- If lookup succeeds, it binds the local queue object to the server queue object.
- If lookup fails, the server can create a new dynamic queue.

Parameter	Description
queue	The function stores the queue object in this location.
queueName	Create a queue with this name. The lookup name of each queue must be unique among all queues.

See Also `tibemsLookupContext` on page 358

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.

Remarks Each queue has a lookup name that is unique among all queues.

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.

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	Page
<code>tibemsSession_DeleteTemporaryQueue</code>	Delete a temporary queue.	325

- Related Types** `tibemsDestination` on page 144
`tibemsQueue` on page 150
- See Also** `tibemsSession_CreateTemporaryQueue` on page 322

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	Page
<code>tibemsSession_DeleteTemporaryTopic</code>	Delete a temporary topic.	326

- Related Types**

`tibemsDestination` on page 144
`tibemsTopic` on page 156
- See Also**

`tibemsSession_CreateTemporaryTopic` on page 323

tibemsTopic

Type


Purpose Topics deliver each message to multiple consumers.

Function	Description	Page
tibemsTopic_Create	Create a topic object.	157
tibemsTopic_Destroy	Destroy a topic object.	158
tibemsTopic_GetTopicName	Get the name of a topic object.	159

Related Types tibemsDestination on page 144
 tibemsTemporaryTopic on page 155

tibemsTopic_Create

Function

Purpose	Create a topic object.
C Declaration	<pre>tibems_status tibemsTopic_Create(tibemsTopic* topic, const char* topicName);</pre>
COBOL Call	<pre>CALL "tibemsTopic_Create" USING BY REFERENCE topic, BY REFERENCE topicName, RETURNING tibems-status END-CALL.</pre>
	topic has usage pointer.
Remarks	<p>This constructor creates only local objects (within the program). It does not attempt to lookup the corresponding server object until the program creates a <code>tibemsMsgConsumer</code> or a <code>tibemsMsgProducer</code> that uses the topic. That automatic lookup can result in either of two outcomes:</p> <ul style="list-style-type: none">• If lookup succeeds, it binds the local topic object to the server topic object.• If lookup fails, the server creates a new dynamic topic.
Parameter	Description
topic	The function stores the topic object in this location.
topicName	Create a topic with this name. The lookup name of each topic must be unique among all topics.
See Also	<code>tibemsLookupContext</code> on page 358

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.

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.

Remarks Each topic has a lookup name that is unique among all topics.

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.

Chapter 5 Consumer

Each message consumer receives messages from a destination.

Topics

- *tibemsMsgConsumer*, page 162
- *tibemsQueueReceiver*, page 171
- *tibemsTopicSubscriber*, page 173

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; related types (such as `tibemsQueueSession` and `tibemsTopicSession`) each define functions to create corresponding consumer subtypes.

Function	Description	Page
<code>tibemsMsgConsumer_Close</code>	Get the message callback and closure data from a consumer.	163
<code>tibemsMsgConsumer_GetMsgListener</code>	Get the message callback and closure data from a consumer.	164
<code>tibemsMsgConsumer_GetMsgSelector</code>	Get the message selector from a consumer.	165
<code>tibemsMsgConsumer_Receive</code>	Receive a message (synchronous).	166
<code>tibemsMsgConsumer_ReceiveNoWait</code>	Receive a message (synchronous, non-blocking).	167
<code>tibemsMsgConsumer_ReceiveTimeout</code>	Receive a message (synchronous, limited blocking).	168
<code>tibemsMsgConsumer_SetMsgListener</code>	Set the message callback and closure data of a consumer.	170

- Related Types** `tibemsQueueReceiver`
`tibemsTopicSubscriber`
- See Also** `tibemsMsgCallback` on page 176
`tibemsSession_CreateConsumer` on page 313

tibemsMsgConsumer_Close

Function

Purpose Stop receiving messages; reclaim resources.

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` on page 166
`tibemsSession_CreateConsumer` on page 313

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	The function stores a pointer to the callback in this location.
	closure	The function stores a pointer to the closure data in this location.

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` on page 170
`tibemsMsgCallback` on page 176

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	The function stores 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 on page 21.

Programs can set this property only when creating the consumer object; see tibemsSession_CreateConsumer on page 313.

See Also tibemsMsgCallback on page 176

tibemsMsgConsumer_Receive

Function

Purpose	Receive a message (synchronous).
C Declaration	<pre>tibems_status tibemsMsgConsumer_Receive(tibemsMsgConsumer msgConsumer, tibemsMsg* message);</pre>
COBOL Call	<pre>CALL "tibemsMsgConsumer_Receive" USING BY VALUE msgConsumer, BY REFERENCE message, RETURNING tibems-status END-CALL.</pre>



msgConsumer and message have usage pointer.

Parameter	Description
msgConsumer	Receive a message through this consumer.
message	The function stores 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 receiving within a transaction, the consumer retains the message until transaction commits.

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.

Parameter	Description
msgConsumer	Receive a message through this consumer.
msg	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 receiving within a transaction, the consumer retains the message until transaction commits.

tibemsMsgConsumer_ReceiveTimeout

Function

Purpose Receive a message (synchronous, limited blocking).

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.

Parameter	Description
msgConsumer	Receive a message through this consumer.
msg	The function stores a pointer to the inbound message in this location.
timeout	When present, 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

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_NOT_FOUND.

When calling receive within a transaction, the consumer retains the message until transaction commits.

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 on page 176

tibemsQueueReceiver

Type

- Purpose

Consume messages from a queue.
- Remarks

The behavior of a queue receiver is almost identical to that of a `tibemsMsgConsumer`, with the exception that it consumes messages only from a queue (not a topic).

Function	Description	Page
<code>tibemsQueueReceiver_GetQueue</code>	Get the queue from a queue receiver.	172

- Related Types

`tibemsMsgConsumer` on page 162

`tibemsSession_CreateConsumer` on page 313

tibemsQueueReceiver_GetQueue

Function

Purpose Get the queue from a queue receiver.

C Declaration

```
tibems_status tibemsQueueReceiver_GetQueue(  
    tibemsQueueReceiver queueReceiver,  
    tibemsQueue* queue );
```

COBOL Call

```
CALL "tibemsQueueReceiver_GetQueue"  
    USING BY VALUE queueReceiver,  
          BY REFERENCE queue,  
          RETURNING tibems-status  
END-CALL.
```



queueReceiver and queue have usage pointer.

Parameters

Parameter	Description
queueReceiver	Get the queue from this receiver.
queue	The function stores the queue in this location.

Remarks The receiver consumes messages from this queue.
Programs set this queue when creating the receiver, and cannot subsequently change it.

tibemsTopicSubscriber

Type

- Purpose** Consume messages from a topic.
- Remarks** The behavior of a topic subscriber is almost identical to that of a `tibemsMsgConsumer`, with the exception that it consumes messages only from a topic (not a queue).

Function	Description	Page
<code>tibemsTopicSubscriber_GetNoLocal</code>	Get the no local property of a topic subscriber.	174
<code>tibemsTopicSubscriber_GetTopic</code>	Get the topic of a topic subscriber.	175

- Related Types**
- `tibemsMsgConsumer` on page 162
 - `tibemsSession_CreateConsumer` on page 313
 - `tibemsSession_CreateDurableSubscriber` on page 315

tibemsTopicSubscriber_GetNoLocal

Function

Purpose Get the no local property of a topic subscriber.

C Declaration

```
tibems_status tibemsTopicSubscriber_GetNoLocal(  
    tibemsTopicSubscriber topicSubscriber,  
    tibems_bool* noLocal );
```

COBOL Call

```
CALL "tibemsTopicSubscriber_GetNoLocal"  
    USING BY VALUE topicSubscriber,  
          BY REFERENCE noLocal,  
          RETURNING tibems-status  
END-CALL.
```



topicSubscriber has usage pointer.

Parameters

Parameter	Description
topicSubscriber	Get the property from this subscriber.
noLocal	The function stores the property value in this location.

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

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

tibemsTopicSubscriber_GetTopic

Function

Purpose Get the topic of a topic subscriber.

C Declaration

```
tibems_status tibemsTopicSubscriber_GetTopic(  
    tibemsTopicSubscriber topicSubscriber,  
    tibemsTopic* topic );
```

COBOL Call

```
CALL "tibemsTopicSubscriber_GetTopic"  
    USING BY VALUE topicSubscriber,  
          BY REFERENCE topic,  
          RETURNING tibems-status  
END-CALL.
```



topicSubscriber and topic have usage pointer.

Parameters

Parameter	Description
topicSubscriber	Get the topic from this subscriber.
topic	The function stores the topic in this location.

Remarks The subscriber consumes messages from this topic.
Programs set this topic property when creating the subscriber, and cannot subsequently change it.

tibemsMsgCallback

Type

Purpose Asynchronously process an arriving message.

C Declaration

```
typedef void (*tibemsMsgCallback) (  
    tibemsMsgConsumer msgConsumer,  
    tibemsMsg msg,  
    void* closure );
```

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.

Parameter	Description
msgConsumer	This parameter receives the consumer object.
msg	This parameter receives the message object.
closure	This parameter receives the closure argument, which your program registered on the consumer.

Serialization In compliance with the JMS specification, sessions distribute messages to consumers in serial (non-concurrent) fashion.

See Also `tibemsMsgConsumer` on page 162

Chapter 6 **Producer**

Message producers send messages to destinations on the server.

Topics

- *tibemsMsgProducer*, page 178
- *tibemsQueueSender*, page 194
- *tibemsTopicPublisher*, page 199

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; related types (such as `tibemsQueueSession` and `tibemsTopicSession`) each define functions to create corresponding producer subtypes.

(Sheet 1 of 2)

Function	Description	Page
<code>tibemsMsgProducer_Close</code>	Destroy the producer object; reclaim resources.	180
<code>tibemsMsgProducer_GetDeliveryMode</code>	Get the delivery mode property of a producer object.	181
<code>tibemsMsgProducer_GetDisableMessageID</code>	Get the disable message ID property of a producer object.	182
<code>tibemsMsgProducer_GetDisableMessageTimestamp</code>	Get the disable message timestamp property of a producer object.	183
<code>tibemsMsgProducer_GetPriority</code>	Get the priority property of a producer object.	184
<code>tibemsMsgProducer_GetTimeToLive</code>	Get the time-to-live property of a producer object.	185
<code>tibemsMsgProducer_Send</code> <code>tibemsMsgProducer_SendEx</code> <code>tibemsMsgProducer_SendToDestination</code> <code>tibemsMsgProducer_SendToDestinationEx</code>	Send a message.	186
<code>tibemsMsgProducer_SetDeliveryMode</code>	Set the delivery mode property of a producer object.	189

(Sheet 2 of 2)

Function	Description	Page
<code>tibemsMsgProducer_SetDisableMessageID</code>	Set the disable message ID property of a producer object.	190
<code>tibemsMsgProducer_SetDisableMessageTimestamp</code>	Set the disable message timestamp property of a producer object.	191
<code>tibemsMsgProducer_SetPriority</code>	Set the priority property of a producer object.	192
<code>tibemsMsgProducer_SetTimeToLive</code>	Set the time-to-live property of a producer object.	193

Related Types `tibemsQueueSender` on page 194
 `tibemsTopicPublisher` on page 199

See Also `tibemsSession_CreateProducer` on page 320

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 on page 320

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	The function stores 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 on page 130.

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
msgProducer	Get the property from this producer.
disable	The function stores 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	The function stores 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_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	The function stores 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	The function stores 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.

Parameter	Description
msgProducer	Send a message through this producer object.
destination	<p>When present, the call sends 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.</p>
message	Send this message object.
deliveryMode	<p>When present, the call sends the message with this delivery mode. This argument is an enumerated value (see <code>tibemsDeliveryMode</code> on page 130). Other send calls send the message with the producer's default delivery mode.</p>
priority	<p>When present, the call sends 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.</p>

Parameter	Description
timeToLive	<p>When present, the call uses this value (in milliseconds) to compute the message expiration.</p> <ul style="list-style-type: none">• 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. <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>

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` on page 130.

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_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.

tibemsQueueSender

Type

Purpose Send messages to a queue.

Function	Description	Page
tibemsQueueSender_GetQueue	Get the queue from a queue sender object.	195
tibemsQueueSender_Send	Send a message.	196
tibemsQueueSender_SendEx		
tibemsQueueSender_SendToQueue		
tibemsQueueSender_SendToQueueEx		

Related Types tibemsMsgProducer on page 178

tibemsQueueSender_GetQueue

Function

Purpose Get the queue from a queue sender object.

C Declaration

```
tibems_status tibemsQueueSender_GetQueue(
    tibemsQueueSender queueSender,
    tibemsQueue* queue );
```

COBOL Call

```
CALL "tibemsQueueSender_GetQueue"
    USING BY VALUE queueSender,
          BY REFERENCE queue,
          RETURNING tibems-status
END-CALL.
```



queueSender and queue have usage pointer.

Parameters

Parameter	Description
queueSender	Get the queue from this sender.
queue	The function stores the queue in this location.

Remarks

Each send call directs a message to a queue.

Programs can use this property to define a default queue for messages that this producer sends. Individual sending calls can override this default value.

Programs set this queue when creating the sender, and cannot subsequently change it.

tibemsQueueSender_Send

Function

Purpose Send a message.

C Declaration

```

tibems_status tibemsQueueSender_Send(
    tibemsQueueSender queueSender,
    tibemsMsg message );

tibems_status tibemsQueueSender_SendEx(
    tibemsQueueSender queueSender,
    tibemsMsg message,
    tibemsDeliveryMode deliveryMode,
    tibems_int priority,
    tibems_long timeToLive );

tibems_status tibemsQueueSender_SendToQueue(
    tibemsQueueSender queueSender,
    tibemsQueue queue,
    tibemsMsg message );

tibems_status tibemsQueueSender_SendToQueueEx(
    tibemsQueueSender queueSender,
    tibemsQueue queue,
    tibemsMsg message,
    tibemsDeliveryMode deliveryMode,
    tibems_int priority,
    tibems_long timeToLive );

```

COBOL Call

```

CALL "tibemsQueueSender_Send"
    USING BY VALUE queueSender,
          BY VALUE message,
          RETURNING tibems-status
END-CALL.

CALL "tibemsQueueSender_SendEx"
    USING BY VALUE queueSender,
          BY VALUE message,
          BY VALUE deliveryMode,
          BY VALUE priority,
          BY VALUE timeToLive,
          RETURNING tibems-status
END-CALL.

CALL "tibemsQueueSender_SendToQueue"
    USING BY VALUE queueSender,
          BY VALUE queue,
          BY VALUE message,
          RETURNING tibems-status
END-CALL.

CALL "tibemsQueueSender_SendToQueueEx"
    USING BY VALUE queueSender,
          BY VALUE queue,

```

```
        BY VALUE message,  
        BY VALUE deliveryMode,  
        BY VALUE priority,  
        BY VALUE timeToLive,  
        RETURNING tibems-status  
END-CALL.
```



queueSender, message and queue have usage pointer.

Parameter	Description
queueSender	Send a message through this queue sender object.
queue	<p>When present, the call sends the message to this queue.</p> <p>Other send calls send the message to the sender’s default queue. When the sender object does not specify a default, the send call must supply this parameter.</p>
message	Send this message object.
deliveryMode	<p>When present, the call sends the message with this delivery mode.</p> <p>This argument is an enumerated value (see tibemsDeliveryMode on page 130).</p> <p>Other send calls send the message with the sender’s default delivery mode.</p>
priority	<p>When present, the call sends the message with this priority.</p> <p>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.</p> <p>Other send calls send the message with the sender’s default priority.</p>

Parameter	Description
timeToLive	<p>When present, the call uses this value (in milliseconds) to compute the message expiration.</p> <ul style="list-style-type: none">• 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. <p>Other send calls use the sender’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>

See Also `tibemsMsgProducer_Send` on page 186

tibemsTopicPublisher

Type

Purpose Send messages to a topic.

Function	Description	Page
tibemsTopicPublisher_Publish	Publish a message to a topic.	201

Related Types tibemsMsgProducer on page 178

tibemsTopicPublisher_GetTopic

Function

Purpose Get the topic property of a topic publisher object.

C Declaration

```
tibems_status tibemsTopicPublisher_GetTopic(  
    tibemsTopicPublisher topicPublisher,  
    tibemsTopic* topic );
```

COBOL Call

```
CALL "tibemsTopicPublisher_GetTopic"  
    USING BY VALUE topicPublisher,  
          BY REFERENCE topic,  
          RETURNING tibems-status  
END-CALL.
```



topicPublisher and topic have usage pointer.

Parameters

Parameter	Description
topicPublisher	Get the property from this publisher.
topic	The function stores the property in this location.

Remarks Each send call directs a message to a topic.

Programs can use this property to define a default topic for messages that this publisher sends. Individual sending calls can override this default value.

Programs set this topic when creating the publisher, and cannot subsequently change it.

tibemsTopicPublisher_Publish

Function

Purpose Publish a message to a topic.

C Declaration

```

tibems_status tibemsTopicPublisher_Publish(
    tibemsTopicPublisher topicPublisher,
    tibemsMsg message );

tibems_status tibemsTopicPublisher_PublishEx(
    tibemsTopicPublisher topicPublisher,
    tibemsMsg message,
    tibemsDeliveryMode deliveryMode,
    tibems_int priority,
    tibems_long timeToLive );

tibems_status tibemsTopicPublisher_PublishToTopic(
    tibemsTopicPublisher topicPublisher,
    tibemsTopic topic,
    tibemsMsg message );

tibems_status tibemsTopicPublisher_PublishToTopicEx(
    tibemsTopicPublisher topicPublisher,
    tibemsTopic topic,
    tibemsMsg message,
    tibemsDeliveryMode deliveryMode,
    tibems_int priority,
    tibems_long timeToLive );

```

COBOL Call

```

CALL "tibemsTopicPublisher_Publish"
    USING BY VALUE topicPublisher,
          BY VALUE message,
          RETURNING tibems-status
END-CALL.

CALL "tibemsTopicPublisher_PublishEx"
    USING BY VALUE topicPublisher,
          BY VALUE message,
          BY VALUE deliveryMode,
          BY VALUE priority,
          BY VALUE timeToLive,
          RETURNING tibems-status
END-CALL.
tibemsTopicPublisher_PublishToTopic

CALL "tibemsTopicPublisher_PublishToTopic"
    USING BY VALUE topicPublisher,
          BY VALUE topic,
          BY VALUE message,
          RETURNING tibems-status
END-CALL.

CALL "tibemsTopicPublisher_PublishToTopicEx"
    USING BY VALUE topicPublisher,

```

```
BY VALUE topic,  
BY VALUE message,  
BY VALUE deliveryMode,  
BY VALUE priority,  
BY VALUE timeToLive,  
RETURNING tibems-status  
END-CALL.
```



topicPublisher, message and topic have usage pointer.

Remarks These calls are parallel to the send calls of tibemsMsgProducer, and they accomplish the same goal—sending messages.

Parameter	Description
topicPublisher	Send a message through this publisher object.
topic	<p>When present, the call sends the message to this topic.</p> <p>Other calls send the message to the publisher’s default topic. When the publisher does not specify a default, the publish call must supply this parameter.</p>
message	Publish this message.
deliveryMode	<p>When present, the call sends the message with this delivery mode.</p> <p>This argument may be either an enumerated value (see tibemsDeliveryMode on page 130) or an integer (see tibemsDeliveryMode on page 130). We recommend enumerated values, because they enable .NET to do stronger type checking at compile time, which can enhance program reliability.</p> <p>Other calls send the message with the publisher’s default delivery mode.</p>
priority	<p>When present, the call sends the message with this priority.</p> <p>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.</p> <p>Other calls send the message with the publisher’s default priority.</p>

Parameter	Description
<code>timeToLive</code>	<p>When present, the call uses this value (in milliseconds) to compute the message expiration.</p> <ul style="list-style-type: none">• 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. <p>Other calls use the publisher'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>

See Also `tibemsMsgProducer_Send` on page 186

Chapter 7 Requestor

Requestors implement convenience functions for request-reply semantics. They send messages (called *requests*) and wait for *reply* messages in response.

Topics

- *tibemsMsgRequestor*, page 206

tibemsMsgRequestor

Type

- Purpose

Encapsulate request-reply semantics.
- Types

tibemsMsgRequestor
tibemsQueueRequestor
tibemsTopicRequestor
- Remarks

tibemsMsgRequestor, tibemsQueueRequestor and tibemsTopicRequestor are all identical types (that is, these names refer to the same underlying struct).

We recommend that programs follow these steps:

 1. Create a tibemsQueueSession or tibemsTopicSession, and use it to create either a tibemsQueue or tibemsTopic (respectively) for requests and replies.
 2. Supply that session and destination to either tibemsQueueRequestor_Create or tibemsTopicRequestor_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	Page
tibemsMsgRequestor_Close	Close a message requestor.	207
tibemsMsgRequestor_Request	Send a request message; wait for a reply.	208
tibemsQueueRequestor_Create	Create a queue requestor.	209
tibemsTopicRequestor_Create	Create a topic requestor.	211

See Also

tibemsQueue on page 150
tibemsTopic on page 156
tibemsQueueSession on page 335
tibemsTopicSession on page 341

tibemsMsgRequestor_Close

Function

Purpose	Close a message requestor.
C Declaration	<pre>tibems_status tibemsMsgRequestor_Close(tibemsMsgRequestor msgRequestor);</pre>
COBOL Call	<pre>CALL "tibemsMsgRequestor_Close" USING BY VALUE msgRequestor, RETURNING tibems-status END-CALL.</pre>



msgRequestor has usage pointer.

Parameter	Description
msgRequestor	Close this requestor.

Remarks This call also closes the requestor’s session as a side effect.

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.

- Remarks

This call blocks indefinitely, until a reply arrives.

The requestor receives only the first reply. It discards other replies that arrive subsequently.

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.

tibemsQueueRequestor_Create

Function

Purpose	Create a queue requestor.
C Declaration	<pre>tibems_status tibemsQueueRequestor_Create(tibemsQueueSession session, tibemsMsgRequestor* msgRequestor, tibemsQueue queue);</pre>
COBOL Call	<pre>CALL "tibemsQueueRequestor_Create" USING BY VALUE session, BY REFERENCE msgRequestor, BY VALUE queue, RETURNING tibems-status END-CALL.</pre>



session, msgRequestor and queue have usage pointer.

Parameter	Description
session	<p>The requestor operates within this queue session.</p> <p>This session must not use transaction semantics. Its delivery mode must be either TIBEMS_AUTO_ACKNOWLEDGE or TIBEMS_DUPS_OK_ACKNOWLEDGE.</p>
msgRequestor	<p>The function stores the new requestor in this location.</p>
queue	<p>The requestor sends request messages to this queue, and waits for replies on the same queue.</p> <p>You <i>must</i> create this queue using the queue session you supply as the first argument.</p>

Remarks	<p>We recommend that programs follow these steps:</p> <ol style="list-style-type: none">1. Create a tibemsQueueSession, and use it to create a tibemsQueue for requests and replies.2. Create a tibemsMsgRequestor, using the queue session and queue 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 `tibemsQueue` on page 150
 `tibemsMsgRequestor_Close` on page 207
 `tibemsMsgRequestor_Request` on page 208
 `tibemsQueueSession` on page 335

tibemsTopicRequestor_Create

Function

Purpose

Create a topic requestor.

C Declaration

```
tibems_status tibemsTopicRequestor_Create(  
    tibemsTopicSession session,  
    tibemsMsgRequestor* msgRequestor,  
    tibemsTopic topic );
```

COBOL Call

```
CALL "tibemsTopicRequestor_Create"  
    USING BY VALUE session,  
          BY REFERENCE msgRequestor,  
          BY VALUE topic,  
          RETURNING tibems-status  
END-CALL.
```



session, msgRequestor and topic have usage pointer.

Parameter	Description
session	<p>The requestor operates within this topic session.</p> <p>This session must not use transaction semantics. Its delivery mode must be either TIBEMS_AUTO_ACKNOWLEDGE or TIBEMS_DUPS_OK_ACKNOWLEDGE.</p>
msgRequestor	<p>The function stores the new requestor in this location.</p>
topic	<p>The requestor sends request messages to this topic, and waits for replies on the same topic.</p> <p>You <i>must</i> create this topic using the topic session you supply as the first argument.</p>

Remarks

We recommend that programs follow these steps:

1. Create a tibemsTopicSession, and use it to create a tibemsTopic for requests and replies.
2. Create a TopicRequestor, using the topic session and topic as arguments.
3. 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.

See Also `tibemsTopic` on page 156
 `tibemsMsgRequestor_Close` on page 207
 `tibemsMsgRequestor_Request` on page 208
 `tibemsTopicSession` on page 341

Chapter 8 **Connection**

Connection objects represent a client program's network connection to the server.

Topics

- *tibemsConnection*, page 214
- *tibemsQueueConnection*, page 229
- *tibemsTopicConnection*, page 233
- *tibemsExceptionCallback*, page 237
- *SSL*, page 238
- *tibemsSSLParams*, page 242
- *tibemsSSLHostNameVerifier*, page 266

tibemsConnection

Type

Purpose	Represent a server connection.
Remarks	<p>When a program first opens a connection, the connection is <i>stopped</i>—that is, it does not deliver inbound messages. To begin the flow of inbound messages, the program must explicitly call <code>tibemsConnection_Start</code>. (Outbound messages flow even before calling <code>tibemsConnection_Start</code>.)</p> <p>The EMS C and COBOL APIs do <i>not</i> support the JMS methods <code>createConnectionConsumer</code> and <code>createDurableConnectionConsumer</code> (which are optional in the JMS specification).</p>
Asynchronous Errors	<p>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.</p> <p>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 <code>tibemsExceptionCallback</code> on page 237).</p>

(Sheet 1 of 2)

Function	Description	Page
<code>tibemsConnection_Close</code>	Close the connection; reclaim resources.	216
<code>tibemsConnection_Create</code> <code>tibemsConnection_CreateSSL</code>	Create a new connection to an EMS server.	218
<code>tibemsConnection_CreateSession</code>	Create a session object.	220
<code>tibemsConnection_GetActiveURL</code>	Get the active URL of a connection.	221
<code>tibemsConnection_GetClientId</code>	Get the client ID of a connection.	222
<code>tibemsConnection_GetExceptionListener</code>	Get the exception listener of a connection.	223
<code>tibemsConnection_SetClientId</code>	Set the client ID of a connection.	224
<code>tibemsConnection_SetExceptionListener</code>	Set the exception listener for a connection.	225
<code>tibemsConnection_Start</code>	Start delivering inbound messages.	226

(Sheet 2 of 2)

Function	Description	Page
tibemsConnection_Stop	Stop delivering inbound messages.	227

Related Types


tibemsQueueConnection on page 229

tibemsTopicConnection on page 233

tibemsConnection_Close

Function

Purpose	Close the connection; reclaim resources.
C Declaration	<pre>tibems_status tibemsConnection_Close(tibemsConnection connection);</pre>
COBOL Call	<pre>CALL "tibemsConnection_Close" USING BY VALUE connection, RETURNING tibems-status END-CALL.</pre>



connection has usage pointer.

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
TIBEMS_ILLEGAL_STATE	The connection is currently processing a message callback.

See Also

tibemsMsg_Acknowledge on page 28

tibemsMsgConsumer on page 162

tibemsMsgProducer on page 178

tibemsDestination on page 144

tibemsSession on page 306

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.

Parameter	Description
connection	The function stores the new connection in this location.
brokerURL	Find the EMS server at this URL.

Parameter	Description
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
TIBEMS_SERVER_NOT_CONNECTED	<ul style="list-style-type: none"> No server is running at the specified URL. The call could not communicate with a server because of mismatched SSL and TCP protocols. Other error situations are possible.
TIBEMS_SECURITY_EXCEPTION	<ul style="list-style-type: none"> The server rejected the connection because the username or password was invalid. SSL setup is incorrect.
TIBEMS_INVALID_CLIENT_ID	The client ID is not unique; that is, another client already uses the ID.

See Also [tibemsQueueConnection_Create](#) on page 230
[tibemsTopicConnection_Create](#) on page 234

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.

Remarks

The new session uses the connection for all server communications.

Parameter	Description
connection	Create a session on this connection.
session	The function stores the new session in this location.
transacted	When true, the new session has transaction semantics. When false, it has non-transaction semantics.
acknowledgeMode	This parameter determines the acknowledge mode of the session. Supply a value enumerated by tibemsAcknowledgeMode.

See Also

tibemsMsg_Acknowledge on page 28

tibemsSession on page 306

tibemsAcknowledgeMode on page 333

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.

Parameter	Description
connection	Get the active URL of this connection.
serverURL	The function stores 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	<pre>tibems_status tibemsConnection_GetClientId(tibemsConnection connection, const char** clientId);</pre>
COBOL Call	<pre>CALL "tibemsConnection_GetClientId" USING BY VALUE connection, BY REFERENCE clientId, RETURNING tibems-status END-CALL.</pre>



connection and clientId have usage pointer.

Parameter	Description
connection	Get the client ID of this connection.
clientId	The function stores a pointer to the ID string in this location.

Remarks	<p>Each connection uses a unique client ID.</p> <p>Client IDs partition the namespace of durable subscribers; see <code>tibemsSession_CreateDurableSubscriber</code> on page 315.</p>
See Also	<code>tibemsConnection_SetClientId</code> on page 224

tibemsConnection_GetExceptionListener

Function

Purpose Get the exception listener of a connection.

C Declaration `tibems_status tibemsConnection_GetExceptionListener(
tibemsConnection connection,
tibemsExceptionCallback* listener,
void** closure);`

Parameter	Description
connection	Get the exception listener of this connection.
listener	The function stores a pointer to the exception listener callback in this location.
closure	The function stores 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` on page 225
Asynchronous Errors on page 214.
`tibemsExceptionCallback` on page 237

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.

Parameter	Description
connection	Set the client ID of this connection.
clientId	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` on page 315.

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` on page 222

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.

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 on page 223
Asynchronous Errors on page 214.
tibemsExceptionCallback on page 237

tibemsConnection_Start

Function

Purpose	Start delivering inbound messages.
C Declaration	<pre>tibems_status tibemsConnection_Start(tibemsConnection connection);</pre>
COBOL Call	<pre>CALL "tibemsConnection_Start" USING BY VALUE connection, RETURNING tibems-status END-CALL.</pre>



connection has usage pointer.

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 on page 227

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.

Parameter	Description
connection	Stop delivering inbound messages on this connection.

Remarks	<p>This call temporarily stops the connection from delivering inbound messages. A program can restart delivery by calling <code>tibemsConnection_Start</code>.</p> <p>When a connection is created, it is stopped. It does not deliver inbound messages until the program calls this function to explicitly start it.</p> <p>If the connection is already stopped, this call has no effect.</p>
Effect	<p>When this call returns, the connection has stopped delivery to all consumers associated with the connection:</p> <ul style="list-style-type: none"> • 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	<p>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.</p> <p>However, the stopped connection prevents the client program from processing any new messages.</p>

Sending A stopped connection can still send outbound messages.

See Also `tibemsConnection_Start` on page 226

tibemsQueueConnection

Type

- Purpose

Backward compatibility. Connection restricted to queue operations.
- Remarks

This type supports existing programs that use it.

For new programs, we recommend using the more general type, `tibemsConnection` on page 214, instead.

Function	Description	Page
<code>tibemsQueueConnection_Create</code> <code>tibemsQueueConnection_CreateSSL</code>	Backward compatibility. Create a connection restricted to queue operations.	230
<code>tibemsQueueConnection_CreateQueueSession</code>	Backward compatibility. Create a queue session object.	232

Related Types

`tibemsConnection` on page 214

tibemsQueueConnection_Create

Function

Purpose Backward compatibility. Create a connection restricted to queue operations.

C Declaration

```
tibems_status tibemsQueueConnection_Create(  
    tibemsQueueConnection* queueConnection,  
    const char* brokerURL,  
    const char* clientId,  
    const char* username,  
    const char* password );  
  
tibems_status tibemsQueueConnection_CreateSSL(  
    tibemsQueueConnection* queueConnection,  
    const char* brokerURL,  
    const char* clientId,  
    const char* username,  
    const char* password,  
    tibemsSSLParams sslParams,  
    const char* pk_password );
```

Parameter	Description
queueConnection	The function stores the new connection in this location.
brokerURL	Find the EMS server at this URL.
clientId	Identify the client program to the server with this unique ID.
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.

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
TIBEMS_SERVER_NOT_CONNECTED	<ul style="list-style-type: none">No server is running at the specified URL.The call could not communicate with a server because of mismatched SSL and TCP protocols.Other error situations are possible.
TIBEMS_SECURITY_EXCEPTION	<ul style="list-style-type: none">The server rejected the connection because the username or password was invalid.SSL setup is incorrect.
TIBEMS_INVALID_CLIENT_ID	The client ID is not unique; that is, another client already uses the ID.

See Also `tibemsConnection_Create` on page 218

tibemsQueueConnection_CreateQueueSession

Function

- Purpose**Backward compatibility. Create a queue session object.
- C Declaration**

```
tibems_status tibemsQueueConnection_CreateQueueSession(  
    tibemsQueueConnection queueConnection,  
    tibemsQueueSession* queueSession,  
    tibems_bool transacted,  
    tibemsAcknowledgeMode acknowledgeMode );
```
- Remarks**The new queue session uses the connection for all server communications.

Parameter	Description
queueConnection	Create a queue session on this connection.
queueSession	The function stores the new session in this location.
transacted	When true, the new session has transaction semantics. When false, it has non-transaction semantics.
acknowledgeMode	This parameter determines the acknowledge mode of the session. Supply a value enumerated by tibemsAcknowledgeMode.

See Also

tibemsMsg_Acknowledge on page 28
tibemsQueueSession on page 335
tibemsAcknowledgeMode on page 333

tibemsTopicConnection

Type

- Purpose**Backward compatibility. Connection restricted to topic operations.
- Remarks**This type supports existing programs that use it.
For new programs, we recommend using the more general type, `tibemsConnection` on page 214, instead.

Function	Description	Page
<code>tibemsTopicConnection_Create</code> <code>tibemsTopicConnection_CreateSSL</code>	Backward compatibility. Create a connection restricted to topic operations.	234
<code>tibemsTopicConnection_CreateTopicSession</code>	Backward compatibility. Create a topic session object.	236

- Related Types**`tibemsConnection` on page 214

tibemsTopicConnection_Create

Function

Purpose Backward compatibility. Create a connection restricted to topic operations.

C Declaration

```
tibems_status tibemsTopicConnection_Create(  
    tibemsTopicConnection* topicConnection,  
    const char* brokerURL,  
    const char* clientId,  
    const char* username,  
    const char* password );  
  
tibems_status tibemsTopicConnection_CreateSSL(  
    tibemsTopicConnection* topicConnection,  
    const char* brokerURL,  
    const char* clientId,  
    const char* username,  
    const char* password,  
    tibemsSSLParams sslParams,  
    const char* pk_password );
```

Parameter	Description
topicConnection	The function stores the new connection in this location.
brokerURL	Find the EMS server at this URL.
clientId	Identify the client program to the server with this unique ID.
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.

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
TIBEMS_SERVER_NOT_CONNECTED	<ul style="list-style-type: none">No server is running at the specified URL.The call could not communicate with a server because of mismatched SSL and TCP protocols.Other error situations are possible.
TIBEMS_SECURITY_EXCEPTION	<ul style="list-style-type: none">The server rejected the connection because the username or password was invalid.SSL setup is incorrect.
TIBEMS_INVALID_CLIENT_ID	The client ID is not unique; that is, another client already uses the ID.

See Also [tibemsConnection_Create](#) on page 218

tibemsTopicConnection_CreateTopicSession

Function

- Purpose

Backward compatibility. Create a topic session object.
- C Declaration

```
tibems_status tibemsTopicConnection_CreateTopicSession(  
    tibemsTopicConnection topicConnection,  
    tibemsTopicSession* topicSession,  
    tibems_bool transacted,  
    tibemsAcknowledgeMode acknowledgeMode );
```
- Remarks

The new topic session uses the connection for all server communications.

Parameter	Description
topicConnection	Create a topic session on this connection.
topicSession	The function stores the new session in this location.
transacted	When true, the new session has transaction semantics. When false, it has non-transaction semantics.
acknowledgeMode	This parameter determines the acknowledge mode of the session. Supply a value enumerated by tibemsAcknowledgeMode.

See Also

tibemsMsg_Acknowledge on page 28

tibemsTopicSession on page 341

tibemsAcknowledgeMode on page 333

tibemsExceptionCallback

Function 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);
```

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.

Parameter	Description
connection	This parameter receives the connection object.
status	This parameter receives a status code, which identifies the connection problem.
closure	This parameter receives the closure data, which the program supplied in the call that registered the callback.

See Also [tibemsConnection](#) on page 214
[tibemsConnection_SetExceptionListener](#) on page 225

SSL

SSL Functions

Function	Description	Page
tibemsSSL_GetTrace	Determine whether SSL tracing is enabled.	239
tibemsSSL_OpenSSLVersion	Get a string representing the OpenSSL version number.	240
tibemsSSL_SetTrace	Enable or disable SSL tracing.	241

SSL Constants

Table 16 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	<pre>tibems_bool tibemsSSL_GetTrace(void); tibems_bool tibemsSSL_GetDebugTrace(void);</pre>
COBOL Call	<pre>CALL "tibemsSSL_GetTrace" RETURNING value-Boolean END-CALL. CALL "tibemsSSL_GetDebugTrace" RETURNING value-Boolean END-CALL.</pre>
Remarks	<p>Two levels of SSL tracing are available—regular tracing and debug tracing (more detailed).</p> <p>If tracing is enabled, these calls return TIBEMS_TRUE.</p> <p>If tracing is disabled, they return TIBEMS_FALSE.</p>

tibemsSSL_OpenSSLVersion

Function

Purpose Get a string representing the OpenSSL version number.

C Declaration

```
const char* tibemsSSL_OpenSSLVersion(  
    char* buffer,  
    tibems_int buf_size );
```

COBOL Call

```
MOVE LENGTH OF buffer TO buf-size.  
  
CALL "tibemsSSL_OpenSSLVersion"  
    USING BY REFERENCE buffer,  
          BY VALUE buf-size,  
          RETURNING value-Pointer  
END-CALL.
```



value-Pointer has usage pointer.

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	<pre>void tibemsSSL_SetTrace(tibems_bool trace); void tibemsSSL_SetDebugTrace(tibems_bool trace);</pre>
COBOL Call	<pre>CALL "tibemsSSL_SetTrace" USING BY VALUE trace END-CALL. CALL "tibemsSSL_SetDebugTrace" USING BY VALUE trace END-CALL.</pre>

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.

(Sheet 1 of 2)

Function	Description	Page
tibemsSSLParams_AddIssuerCert	Add one or more issuer certificates to the SSL parameter object.	244
tibemsSSLParams_AddTrustedCert	Add one or more trusted certificates to the SSL parameter object.	246
tibemsSSLParams_Create	Create a new SSL parameter object.	248
tibemsSSLParams_Destroy	Destroy an SSL parameter object.	249
tibemsSSLParams_GetIdentity	Get the client identity that an SSL parameter object represents.	250
tibemsSSLParams_GetPrivateKey	Get the private key from an SSL parameter object.	251
tibemsSSLParams_SetAuthOnly	Limit the use of SSL to improve performance.	252
tibemsSSLParams_SetCiphers	Set the cipher suites for SSL connections.	253
tibemsSSLParams_SetCRLPath	Set the certificate revocation list (CRL) path.	254
tibemsSSLParams_SetCRLUpdateInterval	Set the certificate revocation list (CRL) update interval.	255
tibemsSSLParams_SetExpectedHostName	Set the expected host name.	256
tibemsSSLParams_SetHostNameVerifier	Set the host name verifier function.	257
tibemsSSLParams_SetIdentity	Set the identity of the client program.	258
tibemsSSLParams_SetPrivateKey	Set the client’s private key.	259

(Sheet 2 of 2)

Function	Description	Page
<code>tibemsSSLParams_SetRandData</code>	Settings for generating random data.	260
<code>tibemsSSLParams_SetRandEGD</code>		
<code>tibemsSSLParams_SetRandFile</code>		
<code>tibemsSSLParams_SetRenegotiateInterval</code>	Set an interval for renegotiating the SSL shared encryption key.	262
<code>tibemsSSLParams_SetRenegotiateSize</code>	Set a size threshold for renegotiating the SSL shared encryption key.	263
<code>tibemsSSLParams_SetVerifyHost</code>	Sets flags that enable client verification of the host certificate or host name.	264

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 );
```

COBOL Call

```
CALL "tibemsSSLParams_AddIssuerCert"  
    USING BY VALUE SSLParams,  
          BY REFERENCE data,  
          BY VALUE size,  
          BY REFERENCE encoding,  
          RETURNING tibems-status  
END-CALL.
```



SSLParams has usage pointer.
COBOL does not support tibemsSSLParams_AddIssuerCertFile in release 4.3.

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 Table 16, Certificate Encodings, on page 238.
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 );
```

COBOL Call

```
CALL "tibemsSSLParams_AddTrustedCert"  
    USING BY VALUE SSLParams,  
          BY REFERENCE data,  
          BY VALUE size,  
          BY VALUE encoding,  
          RETURNING tibems-status  
END-CALL.
```



SSLParams has usage pointer.
COBOL does not support tibemsSSLParams_AddTrustedCertFile in release 4.3.

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 Table 16, Certificate Encodings, on page 238.
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 on page 264), 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);`

COBOL Call `CALL "tibemsSSLParams_Create"
RETURNING SSLParams
END-CALL.`



SSLParams has usage pointer.

tibemsSSLParams_Destroy

Function

- Purpose

Destroy an SSL parameter object.
- C Declaration

void tibemsSSLParams_Destroy(
tibemsSSLParams SSLParams);
- COBOL Call

CALL "tibemsSSLParams_Destroy"
USING BY VALUE SSLParams
END-CALL.



SSLParams has usage pointer.

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 );
```

COBOL Call

```
CALL "tibemsSSLParams_GetIdentity"  
    USING BY VALUE SSLParams,  
          BY REFERENCE data,  
          BY REFERENCE size,  
          BY REFERENCE encoding,  
          RETURNING tibems-status  
END-CALL.
```



SSLParams and data have usage pointer.

Remarks A client identity includes a certificate and private key; it may also include issuer certificates (optional).

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 tibemsSSLParams 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 Table 16, Certificate Encodings, on page 238.

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 );
```

COBOL Call

```
CALL "tibemsSSLParams_GetPrivateKey"
    USING BY VALUE SSLParams,
          BY REFERENCE data,
          BY REFERENCE size,
          BY REFERENCE encoding,
          RETURNING tibems-status
END-CALL.
```



SSLParams and data have usage pointer.

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 tibemsSSLParams 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 Table 16, Certificate Encodings, on page 238.

tibemsSSLParams_SetAuthOnly

Function

Purpose Limit the use of SSL to improve performance.

C Declaration `tibems_status tibemsSSLParams_SetAuthOnly(
tibemsSSLParams SSLParams,
tibems_bool auth_only);`

COBOL Call `CALL "tibemsSSLParams_SetAuthOnly"
USING BY VALUE SSLParams,
BY VALUE auth_only,
RETURNING tibems-status
END-CALL.`



SSLParams has usage pointer.

Remarks For background information, see SSL Authentication Only on page 286 in *TIBCO Enterprise Message Service User's Guide*

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.

tibemsSSLParams_SetCiphers

Function

Purpose Set the cipher suites for SSL connections.

C Declaration

```
tibems_status tibemsSSLParams_SetCiphers(
    tibemsSSLParams SSLParams,
    const char* ciphers );
```

COBOL Call

```
CALL "tibemsSSLParams_SetCiphers"
    USING BY VALUE SSLParams,
          BY REFERENCE ciphers,
    RETURNING tibems-status
END-CALL.
```



SSLParams has usage pointer.

Parameter	Description
SSLParams	Set the value in this SSL parameter object.
ciphers	<p>Specify the cipher suites that the client can use.</p> <p>Supply a colon-separated list of cipher names. Names may be either OpenSSL names, or longer descriptive names.</p> <p>For a list of supported cipher suites, see Supported Cipher Suites on page 283 in <i>TIBCO Enterprise Message Service User's Guide</i>.</p>

tibemsSSLParams_SetCRLPath

Function

- Purpose

Set the certificate revocation list (CRL) path.
- C Declaration

```
tibems_status tibemsSSLParams_SetCRLPath(  
    tibemsSSLParams SSLParams,  
    const char*  crt_path );
```
- COBOL Call

```
CALL "tibemsSSLParams_SetCRLPath"  
    USING BY VALUE SSLParams,  
          BY REFERENCE crt-path,  
          RETURNING tibems-status  
END-CALL.
```



SSLParams has usage pointer.

Parameter	Description
SSLParams	Set the value in this SSL parameter object.
crt_path	<div>The program reads CRL files from this directory.</div> <div>A non-null value for this parameter activates the certificate revocation list (CRL) feature. Specify a directory pathname.</div> <div>Null disables certificate revocation. If your program does not set a value, this is the default behavior.</div>

tibemsSSLParams_SetCRLUpdateInterval

Function

Purpose Set the certificate revocation list (CRL) update interval.

C Declaration `tibems_status tibemsSSLParams_SetCRLUpdateInterval(
tibemsSSLParams SSLParams,
tibems_int hours);`

COBOL Call `CALL "tibemsSSLParams_SetCRLUpdateInterval"
USING BY VALUE SSLParams,
BY VALUE hours,
RETURNING tibems-status
END-CALL.`



SSLParams has usage pointer.

Parameter	Description
SSLParams	Set the value in this SSL parameter object.
hours	The program automatically updates its CRLs at this interval (in hours). When this parameter is left unset, the default is 24 hours.

tibemsSSLParams_SetExpectedHostName

Function

- Purpose

Set the expected host name.
- C Declaration

```
tibems_status tibemsSSLParams_SetExpectedHostName(  
    tibemsSSLParams SSLParams,  
    const char* expected_hostname );
```
- COBOL Call

```
CALL "tibemsSSLParams_SetExpectedHostName"  
    USING BY VALUE SSLParams,  
          BY REFERENCE expected-hostname,  
          RETURNING tibems-status  
END-CALL.
```



SSLParams has usage pointer.

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 on page 257

tibemsSSLParams_SetVerifyHost on page 264

tibemsSSLHostNameVerifier on page 266

tibemsSSLParams_SetHostNameVerifier

Function

Purpose	Set the host name verifier function.
C Declaration	<pre>tibems_status tibemsSSLParams_SetHostNameVerifier(tibemsSSLParams SSLParams, tibemsSSLHostNameVerifier verifier, const void* closure);</pre>
COBOL Call	<pre>CALL "tibemsSSLParams_SetHostNameVerifier" USING BY VALUE SSLParams, BY VALUE verifier, BY REFERENCE closure, RETURNING tibems-status END-CALL.</pre>



SSLParams and verifier have usage pointer.

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	<div>tibemsSSLParams_SetExpectedHostName on page 256</div> <div>tibemsSSLParams_SetVerifyHost on page 264</div> <div>tibemsSSLHostNameVerifier on page 266</div>

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 );
```

COBOL Call

```
CALL "tibemsSSLParams_SetIdentity"  
    USING BY VALUE tibemsSSLParams,  
          BY REFERENCE data,  
          BY VALUE size,  
          BY VALUE encoding,  
          RETURNING tibems-status  
END-CALL.
```



SSLParams has usage pointer.
COBOL does not support tibemsSSLParams_SetIdentityFile in release 4.3.

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 Table 16, Certificate Encodings, on page 238.

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 );
```

COBOL Call

```
CALL "tibemsSSLParams_SetPrivateKey"
    USING BY VALUE SSLParams,
          BY REFERENCE data,
          BY VALUE size,
          BY VALUE encoding,
          RETURNING tibems-status
END-CALL.
```



SSLParams has usage pointer.

COBOL does not support tibemsSSLParams_SetPrivateKeyFile in release 4.3.

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 Table 16, Certificate Encodings, on page 238.

tibemsSSLParams_SetRandData

Function

Purpose Settings for generating random data.

C Declaration

```
tibems_status tibemsSSLParams_SetRandData(  
    tibemsSSLParams SSLParams,  
    const char* rand_data,  
    tibems_int size );  
  
tibems_status tibemsSSLParams_SetRandFile(  
    tibemsSSLParams SSLParams,  
    const char* rand_file );  
  
tibems_status tibemsSSLParams_SetRandEGD(  
    tibemsSSLParams SSLParams,  
    const char* rand_egd_path );
```

COBOL Call

```
CALL "tibemsSSLParams_SetRandData"  
    USING BY VALUE SSLParams,  
          BY REFERENCE rand-data,  
          BY VALUE size,  
          RETURNING tibems-status  
END-CALL.  
  
CALL "tibemsSSLParams_SetRandEGD"  
    USING BY VALUE SSLParams,  
          BY REFERENCE rand-egd-path,  
          RETURNING tibems-status  
END-CALL.
```



SSLParams has usage pointer.
COBOL does not support tibemsSSLParams_SetRandFile in release 4.3.

Remarks These three functions represent three ways to inject crucial random data into SSL computations. Every program must select one of these ways. If an entropy gathering daemon is available on the host computer, we recommend using it.

Parameter	Description
SSLParams	Set the value in this SSL parameter object.
rand_data	Supply random data as a character string.
size	Supply the length (in bytes) of the random data string.
rand_file	Read random data from this file.

Parameter	Description
rand_egd_path	Supply the file pathname of an entropy gathering daemon, which generates random data.

tibemsSSLParams_SetRenegotiateInterval

Function



SSL key renegotiation is deprecated in release 4.3; it is not supported in release 5.0.

Purpose Set an interval for renegotiating the SSL shared encryption key.

C Declaration

```
tibems_status tibemsSSLParams_SetRenegotiateInterval(  
    tibemsSSLParams SSLParams,  
    tibems_long milliseconds );
```

COBOL Call

```
CALL "tibemsSSLParams_SetRenegotiateInterval"  
    USING BY VALUE SSLParams,  
          BY VALUE milliseconds,  
          RETURNING tibems-status  
END-CALL.
```



SSLParams has usage pointer.

Parameter	Description
SSLParams	Set the value in this SSL parameter object.
milliseconds	<p>Set the renegotiation interval (in milliseconds) to this value.</p> <p>The minimum value is 15000 (15 seconds). If you supply a non-zero value that is less than this minimum, this minimum is used instead.</p> <p>Zero is a special value, which disables renegotiation based on time interval.</p>

Remarks The client program renegotiates the key it shares with the server at this interval.



Renegotiating a key can adversely affect overall performance. If you set renegotiation parameters, ensure that renegotiation occurs only when truly required.

tibemsSSLParams_SetRenegotiateSize

Function



SSL key renegotiation is deprecated in release 4.3; it is not supported in release 5.0.

Purpose Set a size threshold for renegotiating the SSL shared encryption key.

C Declaration

```
tibems_status tibemsSSLParams_SetRenegotiateSize(  
    tibemsSSLParams SSLParams,  
    tibems_long bytes );
```

COBOL Call

```
CALL "tibemsSSLParams_SetRenegotiateSize"  
    USING BY VALUE SSLParams,  
          BY VALUE bytes,  
          RETURNING tibems-status  
END-CALL.
```



SSLParams has usage pointer.

Parameter	Description
SSLParams	Set the value in this SSL parameter object.
bytes	<p>Set the renegotiation size (in bytes) to this value.</p> <p>The minimum value is 65536. If you supply a non-zero value that is less than this minimum, this minimum is used instead.</p> <p>Zero is a special value, which disables renegotiation based on data size.</p>

Remarks The client program renegotiates the key it shares with the server after encrypting this number of bytes.



Renegotiating a key can adversely affect overall performance. If you set renegotiation parameters, ensure that renegotiation occurs only when truly required.

tibemsSSLParams_SetVerifyHost

Function

Purpose Sets 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 params,  
    tibems_bool verify );
```

COBOL Call

```
CALL "tibemsSSLParams_SetVerifyHost"  
  USING BY VALUE SSLParams,  
         BY VALUE verify,  
         RETURNING tibems-status  
END-CALL.  
  
CALL "tibemsSSLParams_SetVerifyHostName"  
  USING BY VALUE SSLParams,  
         BY VALUE verify,  
         RETURNING tibems-status  
END-CALL.
```



SSLParams has usage pointer.

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 on page 246).

tibemsSSLParams_SetVerifyHostName enables checking the server’s actual host name against an expected server host name; see tibemsSSLParams_SetExpectedHostName on page 256.

Parameter	Description
SSLParams	Set the value in this SSL parameter object.
verify	TIBEMS_TRUE enables verification. TIBEMS_FALSE disables verification.

See Also tibemsSSLParams_SetHostNameVerifier on page 257

`tibemsSSLHostNameVerifier` on page 266

tibemsSSLHostNameVerifier

Function 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 );
```

Parameter	Description
connected_hostname	This parameter receives the actual host name of the server to which the client program is attempting to connect.
expected_hostname	This parameter receives the host name that the client expects the server to be running on.
certificate_name	This parameter receives the host name in the server’s public certificate.
closure	This parameter receives 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 on page 256

`tibemsSSLParams_SetHostNameVerifier` on page 257
`tibemsSSLParams_SetVerifyHost` on page 264

Chapter 9 Connection Factory

Connection factories let administrators preconfigure client connections to the EMS server.

Topics

- *tibemsConnectionFactory*, page 270
- *tibemsQueueConnectionFactory*, page 297
- *tibemsTopicConnectionFactory*, page 301

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 tibemsd 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.
Separate JNDI Repository	Administrators define factories in a JNDI repository. Client programs call tibemsLookupContext_Lookup to retrieve factories, and use them to create connections to the server.

(Sheet 1 of 3)

Function	Description	Page
tibemsConnectionFactory_Create	Create a connection factory.	273
tibemsConnectionFactory_CreateConnection	Create a connection object.	274
tibemsConnectionFactory_CreateConnectionSSL	Create an SSL connection object.	275
tibemsConnectionFactory_Destroy	Destroy a connection factory object.	277
tibemsConnectionFactory_GetSSLProxyHost	Get the SSL proxy host from a connection factory.	278
tibemsConnectionFactory_GetSSLProxyPort	Get the SSL proxy port from a connection factory.	279

(Sheet 2 of 3)

Function	Description	Page
<code>tibemsConnectionFactory_GetSSLProxyUser</code>	Get the SSL proxy username from a connection factory.	280
<code>tibemsConnectionFactory_GetSSLProxyPassword</code>	Get the SSL proxy password from a connection factory.	281
<code>tibemsConnectionFactory_GetType</code>	Get the type of a connection factory object.	282
<code>tibemsConnectionFactory_SetClientID</code>	Set the client ID of a connection factory object.	283
<code>tibemsConnectionFactory_SetConnectAttemptCount</code>	Modify the connection attempts setting.	284
<code>tibemsConnectionFactory_SetConnectAttemptDelay</code>	Modify the connection delay setting.	285
<code>tibemsConnectionFactory_SetMetric</code>	Modify the load balancing metric.	286
<code>tibemsConnectionFactory_SetReconnectAttemptCount</code>	Modify the reconnection attempts setting.	287
<code>tibemsConnectionFactory_SetReconnectAttemptDelay</code>	Modify the reconnection delay setting.	288
<code>tibemsConnectionFactory_SetServerURL</code>	Set the server URL.	289
<code>tibemsConnectionFactory_SetSSLParams</code>	Set a connection factory's default SSL parameters.	290
<code>tibemsConnectionFactory_SetSSLProxy</code>	Set a connection factory's parameters for connecting through an SSL proxy.	291
<code>tibemsConnectionFactory_SetSSLProxyAuth</code>	Set a connection factory's username and password for connecting through an SSL proxy.	293

(Sheet 3 of 3)

Function	Description	Page
tibemsConnectionFactory_SetType	Set the type of a connection factory object.	294
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.	
Related Types	tibemsQueueConnectionFactory on page 297 tibemsTopicConnectionFactory on page 301	
See Also	tibemsLookupContext on page 358	

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` on page 358).

See Also `tibemsLookupContext` on page 358

tibemsConnectionFactory_CreateConnection

Function

Purpose	Create a connection object.
C Declaration	<pre>tibems_status tibemsConnectionFactory_CreateConnection(tibemsConnectionFactory factory, tibemsConnection* connection, const char * username, const char * password);</pre>
COBOL Call	<pre>CALL "tibemsConnectionFactory_CreateConnection" USING BY VALUE factory, BY REFERENCE connection, BY REFERENCE username, BY REFERENCE password, RETURNING tibems-status END-CALL.</pre>



factory and connection have usage pointer.

Parameter	Description
factory	Use this connection factory to create a connection.
connection	The function stores the new connection object in this location.
userName	When non-null, the connection object presents this user identity to the server.
password	When non-null, the connection object authenticates the user identity with this password.

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 on page 214

tibemsConnectionFactory_CreateConnectionSSL

Function

Purpose Create an SSL connection object.

C Declaration

```
tibems_status tibemsConnectionFactory_CreateConnectionSSL(  
    tibemsConnectionFactory factory,  
    tibemsConnection* connection,  
    const char * username,  
    const char * password,  
    tibemsSSLParams SSLparams,  
    const char * pk_password );
```

COBOL Call

```
CALL "tibemsConnectionFactory_CreateConnectionSSL"  
    USING BY VALUE factory,  
          BY REFERENCE connection,  
          BY REFERENCE username,  
          BY REFERENCE password,  
          BY VALUE SSLparams,  
          BY REFERENCE pk-password,  
          RETURNING tibems-status  
END-CALL.
```



factory, connection, and SSLparams have usage pointer.

Parameter	Description
factory	Use this connection factory to create a connection.
connection	The function stores the new connection object in this location.
userName	When non-null, the connection object presents this user identity to the server.
password	When non-null, the connection object authenticates the user identity with this password.
SSLParams	When non-null, the connection establishes SSL communication using these parameters, overriding the connection factory's default SSL parameters. Null is a special value, which instructs the call to use the connection factory's default SSL parameters.
pk_password	Private key password for SSL.

- 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

tibemsSSLParams on page 242

tibemsConnection on page 214

tibemsConnectionFactory_Destroy

Function

Purpose	Destroy a connection factory object.
C Declaration	<pre>tibems_status tibemsConnectionFactory_Destroy(tibemsConnectionFactory factory)</pre>
COBOL Call	<pre>CALL "tibemsConnectionFactory_Destroy" USING BY VALUE factory, RETURNING tibems-status END-CALL.</pre>
	factory has usage pointer.
Remarks	This call destroys an object within the program. It does not affect objects in the repository.
Parameter	Description
factory	Destroy this connection factory.

tibemsConnectionFactory_GetSSLProxyHost

Function

- Purpose**Get the SSL proxy host from a connection factory.
- C Declaration**

```
tibems_status tibemsConnectionFactory_GetSSLProxyHost(  
    tibemsAnyConnectionFactory 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.

Parameter	Description
factory	Get the SSL proxy host from this connection factory. You may supply any type of connection factory, and cast it to tibemsAnyConnectionFactory.
proxy_host	The function stores the proxy host in this location.

See Also tibemsConnectionFactory_SetSSLProxy on page 291

tibemsConnectionFactory_GetSSLProxyPort

Function

Purpose Get the SSL proxy port from a connection factory.

C Declaration `tibems_status tibemsConnectionFactory_GetSSLProxyPort(
tibemsAnyConnectionFactory 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.

Parameter	Description
factory	Get the SSL proxy port number from this connection factory. You may supply any type of connection factory, and cast it to tibemsAnyConnectionFactory.
proxy_port	The function stores the proxy port in this location.

See Also tibemsConnectionFactory_SetSSLProxy on page 291

tibemsConnectionFactory_GetSSLProxyUser

Function

- Purpose

Get the SSL proxy username from a connection factory.
- C Declaration

```
tibems_status tibemsConnectionFactory_GetSSLProxyUser(  
    tibemsAnyConnectionFactory 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.

Parameter	Description
factory	Get the SSL proxy username from this connection factory. You may supply any type of connection factory, and cast it to tibemsAnyConnectionFactory.
proxy_user	The function stores the proxy user name in this location.

See Also

tibemsConnectionFactory_SetSSLProxyAuth on page 293

tibemsConnectionFactory_GetSSLProxyPassword

Function

Purpose Get the SSL proxy password from a connection factory.

C Declaration `tibems_status tibemsConnectionFactory_GetSSLProxyPassword(
tibemsAnyConnectionFactory 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.

Parameter	Description
factory	Get the SSL proxy password from this connection factory. You may supply any type of connection factory, and cast it to tibemsAnyConnectionFactory.
proxy_password	The function stores the proxy password in this location.

See Also tibemsConnectionFactory_SetSSLProxyAuth on page 293

tibemsConnectionFactory_GetType

Function

- Purpose**Get the type of a connection factory object.
- C Declaration**

```
tibems_status tibemsConnectionFactory_GetType(  
    tibemsConnectionFactory factory,  
    tibemsConnectionFactoryType* type );
```
- COBOL Call**

```
CALL "tibemsConnectionFactory_GetType"  
    USING BY VALUE factory,  
          BY REFERENCE type,  
          RETURNING tibems-status  
END-CALL.
```



factory has usage pointer.

Parameter	Description
factory	Get the type of this connection factory.
type	The function stores the type indicator in this location.

See Also tibemsConnectionFactoryType on page 295

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.

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.

Parameter	Description
factory	Set the connection attempts parameter of this connection factory.
connAttempts	<div>This value limits the number of times that a connection object attempts to establish a connection to the server. The minimum value is 1.</div> <div>When this property is not set, the default value is 2.</div>

See Also

tibemsConnectionFactory_SetConnectAttemptDelay on page 285

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.

Parameter	Description
factory	Set the connection delay parameter of this connection factory.
delay	<div>This value determines the time (in milliseconds) between connection attempts. The minimum value is 250.</div> <div>When this property is not set, the default value is 500.</div>

See Also

tibemsConnectionFactory_SetConnectAttemptCount on page 284

tibemsConnectionFactory_SetMetric

Function

Purpose	Modify the load balancing metric.
C Declaration	<pre>tibems_status tibemsConnectionFactory_SetMetric(tibemsConnectionFactory factory, tibemsFactoryLoadBalanceMetric metric);</pre>
COBOL Call	<pre>CALL "tibemsConnectionFactory_SetMetric" USING BY VALUE factory, BY VALUE metric, RETURNING tibems-status END-CALL.</pre>



factory has usage pointer.

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 on page 296.
---------	--

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.

Remarks

tibemsConnection objects attempt to reconnect to the server after a network disconnect.

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. See tibemsConnectionFactory_SetServerURL on page 289.

Parameter	Description
factory	Set the reconnection attempts parameter of this connection factory.
connAttempts	<p>This value limits the number of times that a connection object attempts to reestablish a connection to the server. The minimum value is 1.</p> <p>When this property is not set, the default value is 4.</p>

See Also

tibemsConnectionFactory_SetReconnectAttemptDelay on page 288

tibemsConnectionFactory_SetReconnectAttemptDelay

Function

Purpose	Modify the reconnection delay setting.
C Declaration	<pre>tibems_status tibemsConnectionFactory_SetReconnectAttemptDelay(tibemsConnectionFactory factory, tibems_int delay);</pre>
COBOL Call	<pre>CALL "tibemsConnectionFactory_SetReconnectAttemptDelay" USING BY VALUE factory, BY VALUE delay, RETURNING tibems-status END-CALL.</pre>



factory has usage pointer.

Remarks tibemsConnectionFactory objects attempt to reconnect to the server after a network disconnect.

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. See tibemsConnectionFactory_SetServerURL on page 289.

Parameter	Description
factory	Set the reconnection delay parameter of this connection factory.
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 on page 287

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.

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).

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.

Parameter	Description
factory	Set the default SSL parameters of this connection factory.
sslParams	The connection establishes SSL communication using these parameters.

See Also tibemsSSLParams on page 242
tibemsConnectionFactory_CreateConnectionSSL on page 275

tibemsConnectionFactory_SetSSLProxy

Function

Purpose Set a connection factory’s parameters for connecting through an SSL proxy.

C Declaration `tibems_status tibemsConnectionFactory_SetSSLProxy(
tibemsAnyConnectionFactory 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.

Parameter	Description
factory	Set the SSL proxy host and port on this connection factory. You may supply any type of connection factory, and cast it to tibemsAnyConnectionFactory.
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).
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 on page 278

`tibemsConnectionFactory_GetSSLProxyPort` on page 279
`tibemsConnectionFactory_SetSSLProxyAuth` on page 293

tibemsConnectionFactory_SetSSLProxyAuth

Function

Purpose Set a connection factory’s username and password for connecting through an SSL proxy.

C Declaration

```
tibems_status
tibemsConnectionFactory_SetSSLProxyAuth(
    tibemsAnyConnectionFactory 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.

Parameter	Description
factory	Set the username and password on this connection factory. You may supply any type of connection factory, and cast it to tibemsAnyConnectionFactory.
proxy_user	The connection factory authenticates itself to the SSL proxy using this username.
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 on page 280
tibemsConnectionFactory_GetSSLProxyPassword on page 281
tibemsConnectionFactory_SetSSLProxy on page 291

tibemsConnectionFactory_SetType

Function

Purpose	Set the type of a connection factory object.
C Declaration	<pre>tibems_status tibemsConnectionFactory_SetType(tibemsConnectionFactory factory, tibemsConnectionFactoryType type);</pre>
COBOL Call	<pre>CALL "tibemsConnectionFactory_SetType" USING BY VALUE factory, BY VALUE type, RETURNING tibems-status END-CALL.</pre>



factory has usage pointer.

Parameter	Description
factory	Set the type of this connection factory.
type	Set the indicator to this type.

See Also tibemsConnectionFactoryType on page 295

tibemsConnectionFactoryType

Type

Purpose Indicate the type of a connection factory, and guide the type of connection it creates.

Constants

- TIBEMS_CONNECTION_FACTORY
- TIBEMS_QUEUE_CONNECTION_FACTORY
- TIBEMS_TOPIC_CONNECTION_FACTORY
- TIBEMS_XA_CONNECTION_FACTORY
- TIBEMS_XA_QUEUE_CONNECTION_FACTORY
- TIBEMS_XA_TOPIC_CONNECTION_FACTORY

COBOL * NOTE: FAC is the shortened name for FACTORY
01 TIBEMS-CONNECTIONFACTORY-TYPES.
 05 tibemsConnectionFactoryType PIC S9(8) BINARY.
 88 TIBEMS-CONNECTION-FAC VALUE 0.
 88 TIBEMS-QUEUE-CONNECTION-FAC VALUE 1.
 88 TIBEMS-TOPIC-CONNECTION-FAC VALUE 2.
 88 TIBEMS-XA-CONNECTION-FAC VALUE 3.
 88 TIBEMS-XA-QUEUE-CONNECTION-FAC VALUE 4.
 88 TIBEMS-XA-TOPIC-CONNECTION-FAC VALUE 5.

See Also tibemsConnectionFactory_GetType on page 282
 tibemsConnectionFactory_SetType on page 294

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	Indicates 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.
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 on page 270

tibemsConnectionFactory_SetMetric on page 286

tibemsQueueConnectionFactory

Type

Purpose Backward compatibility. Administered object for creating queue connections.

Function	Description	Page
<code>tibemsQueueConnectionFactory_CreateConnection</code>	Backward compatibility. Create a queue connection object.	298
<code>tibemsQueueConnectionFactory_CreateConnectionSSL</code>	Backward compatibility. Create an SSL queue connection object.	299

Related Types `tibemsConnectionFactory` on page 270

tibemsQueueConnectionFactory_CreateConnection

Function

Purpose Backward compatibility. Create a queue connection object.

C Declaration

```
tibems_status tibemsQueueConnectionFactory_CreateConnection(  
    tibemsQueueConnectionFactory factory,  
    tibemsQueueConnection* connection,  
    const char * username,  
    const char * password );
```

COBOL Call

```
CALL "tibemsQueueConnectionFactory_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.

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.

Parameter	Description
factory	Use this connection factory to create a connection.
connection	The function stores the new connection object in this location.
userName	When non-null, the connection object presents this user identity to the server.
password	When non-null, the connection object authenticates the user identity with this password.

See Also tibemsQueueConnection on page 229

tibemsQueueConnectionFactory_CreateConnectionSSL

Function

Purpose	Backward compatibility. Create an SSL queue connection object.
C Declaration	<pre>tibems_status tibemsQueueConnectionFactory_CreateConnectionSSL(tibemsQueueConnectionFactory factory, tibemsQueueConnection* connection, const char * username, const char * password, tibemsSSLParams SSLparams, const char * pk_password);</pre>
COBOL Call	<pre>CALL "tibemsQueueConnectionFactory_CreateConnectionSSL" USING BY VALUE factory, BY REFERENCE connection, BY REFERENCE username, BY REFERENCE password, BY VALUE SSLparams, BY REFERENCE pk-password, RETURNING tibems-status END-CALL.</pre>



factory, connection and SSLParams have usage pointer.

Parameter	Description
factory	Use this connection factory to create a connection.
connection	The function stores the new connection object in this location.
userName	When non-null, the connection object presents this user identity to the server.
password	When non-null, the connection object authenticates the user identity with this password.
SSLParams	The connection establishes SSL communication using these parameters.
pk_password	Private key password for SSL.

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	tibemsSSLParams on page 242

`tibemsQueueConnection` on page 229

tibemsTopicConnectionFactory

Type

Purpose Backward compatibility. Administered object for creating topic connections.

Function	Description	Page
tibemsTopicConnectionFactory_CreateConnection	Backward compatibility. Create a topic connection object.	302
tibemsTopicConnectionFactory_CreateConnectionSSL	Backward compatibility. Create an SSL topic connection object.	303

Related Types tibemsConnectionFactory on page 270

tibemsTopicConnectionFactory_CreateConnection

Function

Purpose	Backward compatibility. Create a topic connection object.
C Declaration	<pre>tibems_status tibemsTopicConnectionFactory_CreateConnection(tibemsTopicConnectionFactory factory, tibemsTopicConnection* connection, const char * username, const char * password);</pre>
COBOL Call	<pre>CALL "tibemsTopicConnectionFactory_CreateConnection" USING BY VALUE factory, BY REFERENCE connection, BY REFERENCE username, BY REFERENCE password, RETURNING tibems-status END-CALL.</pre>



factory and connection have usage pointer.

Parameter	Description
factory	Use this connection factory to create a connection.
connection	The function stores the new connection object in this location.
userName	When non-null, the connection object presents this user identity to the server.
password	When non-null, the connection object authenticates the user identity with this password.

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 tibemsTopicConnection on page 233

tibemsTopicConnectionFactory_CreateConnectionSSL

Function

Purpose Backward compatibility. Create an SSL topic connection object.

C Declaration

```
tibems_status tibemsTopicConnectionFactory_CreateConnectionSSL(  
    tibemsTopicConnectionFactory factory,  
    tibemsTopicConnection* connection,  
    const char * username,  
    const char * password,  
    tibemsSSLParams sslParams,  
    const char * pk_password );
```

COBOL Call

```
CALL "tibemsTopicConnectionFactory_CreateConnectionSSL"  
  USING BY VALUE factory,  
        BY REFERENCE connection,  
        BY REFERENCE username,  
        BY REFERENCE password,  
        BY VALUE sslParams,  
        BY REFERENCE pk-password,  
        RETURNING tibems-status  
END-CALL.
```



factory, connection and sslParams have usage pointer.

Parameter	Description
factory	Use this connection factory to create a connection.
connection	The function stores the new connection object in this location.
userName	When non-null, the connection object presents this user identity to the server.
password	When non-null, the connection object authenticates the user identity with this password.
sslParams	The connection establishes SSL communication using these parameters.
pk_password	Private key password for SSL.

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 tibemsSSLParams on page 242

`tibemsTopicConnection` on page 233

Chapter 10 **Session**

A session is a single-threaded context for producing and consuming messages.

Topics

- *tibemsSession*, page 306
- *tibemsAcknowledgeMode*, page 333
- *tibemsQueueSession*, page 335
- *tibemsTopicSession*, page 341

tibemsSession

Type

Purpose	Organizing context for message activity.
Remarks	<div>Sessions combine several roles:<ul style="list-style-type: none">• 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).</div>
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 <code>tibemsMsgConsumer_Receive</code>) 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.

(Sheet 1 of 2)

Function	Description	Page
Messages		
<code>tibemsSession_CreateBytesMessage</code>	Create a byte array message.	312
<code>tibemsSession_CreateMapMessage</code>	Create a map message.	318
<code>tibemsSession_CreateStreamMessage</code>	Create a stream message.	321
<code>tibemsSession_CreateTextMessage</code>	Create a text message.	324
Destinations		
<code>tibemsSession_CreateBrowser</code>	Create a queue browser.	311
<code>tibemsSession_CreateTemporaryQueue</code>	Create a temporary queue.	322
<code>tibemsSession_CreateTemporaryTopic</code>	Create a temporary topic.	323
Consumers & Producers		
<code>tibemsSession_CreateConsumer</code>	Create a message consumer.	313
<code>tibemsSession_CreateDurableSubscriber</code>	Create a durable topic subscriber.	315

(Sheet 2 of 2)

Function	Description	Page
tibemsSession_CreateProducer	Create a message producer.	320
tibemsSession_Unsubscribe	Unsubscribe a durable topic subscription.	332
Transactions		
tibemsSession_Commit	Commit the open transaction.	310
tibemsSession_Rollback	Roll back messages in the current transaction.	331
Other		
tibemsSession_Close	Close a session; reclaim resources.	309
tibemsSession_Recover	Recover from undetermined state during message processing.	329

Related Types

tibemsQueueSession on page 335

tibemsTopicSession on page 341

tibemsSession_Close

Function

Purpose	Close a session; reclaim resources.
C Declaration	<pre>tibems_status tibemsSession_Close(tibemsSession session);</pre>
COBOL Call	<pre>CALL "tibemsSession_Close" USING BY VALUE session, RETURNING tibems-status END-CALL.</pre>



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.

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 Message Selectors on page 21. When null, or the empty string, the browser views all messages in the queue.

See Also tibemsQueue on page 150
 tibemsQueueBrowser on page 352

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.

Parameter	Description
session	Create the message in this session.
bytesMsg	The function stores the new message object in this location.

See Also tibemsBytesMsg on page 76

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.

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	<p>When non-null, the server filters messages using this selector, so the consumer receives only matching messages; see Message Selectors on page 21.</p> <p>When null, or the empty string, the consumer receives messages without filtering.</p>
noLocal	<p>When true, the server filters messages so the consumer 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>

See Also `tibemsDestination` on page 144
 `tibemsMsgConsumer` on page 162

tibemsSession_CreateDurableSubscriber

Function

Purpose Create a durable topic subscriber.

C Declaration

```
tibems_status tibemsSession_CreateDurableSubscriber(
    tibemsSession session,
    tibemsTopicSubscriber* topicSubscriber,
    tibemsTopic topic,
    const char* name,
    const char* messageSelector,
    tibems_bool noLocal );
```

COBOL Call

```
CALL "tibemsSession_CreateDurableSubscriber"
    USING BY VALUE session,
          BY REFERENCE topicSubscriber,
          BY VALUE topic,
          BY REFERENCE name,
          BY REFERENCE messageSelector,
          BY VALUE noLocal,
          RETURNING tibems-status
END-CALL.
```



session, subscriber and topic have usage pointer.

Parameter	Description
session	Create the topic subscriber in this session.
topicSubscriber	The function stores the new topic subscriber object in this location.
topic	Create a durable subscriber for this topic (which <i>cannot</i> be a tibemsTemporaryTopic).
name	This unique name lets the server associate the subscriber with a subscription.
messageSelector	When non-null, the server filters messages using this selector, so the subscriber receives only matching messages; see Message Selectors on page 21. When null, or the empty string, the subscriber receives messages without filtering.

Parameter	Description
noLocal	<p>When <code>true</code>, 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 <code>false</code>, the consumer receives messages with local origin.</p>
Remarks	<p>The server associates a durable subscription with at most one subscriber object at a time. When a subscriber object exists, the subscription is <i>active</i>, and the server delivers messages to it; when no subscriber object exists, the subscription is <i>inactive</i>.</p> <p>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.</p>
Subscription Continuity	<p>Continuity across inactive periods uses two data items from the client:</p> <ul style="list-style-type: none">• Subscription Name a parameter of this call• Client ID an optional property of the <code>tibemsConnection</code> (used only when supplied) <p>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.</p> <p>If a matching subscription exists, but it is already active, this function fails with <code>TIBEMS_INVALID_CONSUMER</code>.</p> <p>If a matching subscription to the topic does not yet exist, the server creates one.</p>
Matching Client ID	<ul style="list-style-type: none">• If the <code>tibemsConnection</code>’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 <code>tibemsConnection</code>’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	<p>Notice that the server does <i>not</i> use the topic and message selector arguments to match a subscriber to an existing subscription. As a result, client programs can <i>change</i> 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).</p>

See Also [tibemsTopic](#) on page 156
 [tibemsTopicSubscriber](#) on page 173
 [tibemsConnection](#) on page 214

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.

Remarks The JMS specification requires this call. It is equivalent to tibemsMapMsg_Create on page 92.

Parameter	Description
session	Create the message in this session.
mapMsg	The function stores the new message object in this location.

See Also tibemsMapMsg on page 91

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.

Remarks The JMS specification requires this call. It is equivalent to tibemsMsg_Create on page 32.

Parameter	Description
session	Create the message in this session.
message	The function stores the new message object in this location.

See Also tibemsMsg on page 25

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.

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 on page 144
 tibemsMsgProducer on page 178

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.

Remarks The JMS specification requires this call. It is equivalent to tibemsStreamMsg_Create on page 112.

Parameter	Description
session	Create the message in this session.
streamMsg	The function stores the new message object in this location.

See Also tibemsStreamMsg on page 110

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.

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 on page 154

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.

Parameter	Description
session	Create the queue 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` on page 155

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.

Remarks The JMS specification requires these calls. It is equivalent to tibemsTextMsg_Create on page 126.

Parameter	Description
session	Create the message in this session.
textMsg	The function stores the new message object in this location.
text	Create a text message with this text as its body.

See Also tibemsTextMsg on page 125

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.

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.
- See Also

tibemsTemporaryQueue on page 154

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.

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.
- See Also** tibemsTemporaryTopic on page 155

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.

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` on page 333.

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.

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	<pre>tibems_status tibemsSession_Recover(tibemsSession session);</pre>
COBOL Call	<pre>CALL "tibemsSession_Recover" USING BY VALUE session, RETURNING tibems-status END-CALL.</pre>



session has usage pointer.

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 do this sequence of actions:

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 When a session has transactional semantics, this call is inappropriate, and returns `TIBEMS_INVALID_SESSION` (commit and rollback are more appropriate with transactions).

tibemsSession_Rollback

Function

Purpose	Roll back messages in the current transaction.
C Declaration	<pre>tibems_status tibemsSession_Rollback(tibemsSession session);</pre>
COBOL Call	<pre>CALL "tibemsSession_Rollback" USING BY VALUE session, RETURNING tibems-status END-CALL.</pre>



session has usage pointer.

Remarks	When a session does not have transactional semantics, this function returns TIBEMS_ILLEGAL_STATE.
---------	---

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.

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` or `tibemsTopicSubscriber` exists.

It is illegal to delete a subscription while one of its messages is either unacknowledged, or uncommitted (in the current transaction).xxxxxxx

Parameter	Description
session	Delete a subscription in this session.
name	This name lets the server locate the subscription.

See Also `tibemsMsgConsumer` on page 162
`tibemsTopicSubscriber` on page 173
`tibemsTopic` on page 156
`tibemsSession_CreateDurableSubscriber` on page 315

tibemsAcknowledgeMode

Type

Purpose Define acknowledgement mode constants.

(Sheet 1 of 2)

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">• synchronous receive calls (such as <code>tibemsMsgConsumer_Receive</code> on page 166)• asynchronous listener callback (namely, <code>tibemsMsgCallback</code> on page 176)
TIBEMS_CLIENT_ACKNOWLEDGE	<p>In this mode, the client program acknowledges receipt by calling <code>tibemsMsg_Acknowledge</code> on page 28. Each call acknowledges all messages received so far.</p>
TIBEMS_DUPS_OK_ACKNOWLEDGE	<p>As with <code>TIBEMS_AUTO_ACKNOWLEDGE</code>, the session automatically acknowledges messages. However, it may do so lazily.</p> <p><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.</p>

(Sheet 2 of 2)

Constant	Description
TIBEMS_EXPLICIT_CLIENT_ACKNOWLEDGE	<p>As with TIBEMS_CLIENT_ACKNOWLEDGE, the client program acknowledges receipt by calling <code>tibemsMsg_Acknowledge</code> on page 28. However, each call acknowledges <i>only</i> the individual message. The client may acknowledge messages in any order.</p> <p>This mode and behavior are proprietary extensions, specific to TIBCO EMS.</p>
TIBEMS_EXPLICIT_CLIENT_DUPS_OK_ACKNOWLEDGE	<p>In this mode, the client program lazily acknowledges <i>only</i> the individual message, by calling <code>tibemsMsg_Acknowledge</code> on page 28. The client may acknowledge messages in any order.</p> <p><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.</p> <p>This mode and behavior are proprietary extensions, specific to TIBCO EMS.</p>
TIBEMS_NO_ACKNOWLEDGE	<p>In TIBEMS_NO_ACKNOWLEDGE mode, messages do not require acknowledgement (which reduces message overhead). The server never redelivers messages.</p> <p>This mode is available for <i>topic</i> sessions only.</p> <p>This mode and behavior are proprietary extensions, specific to TIBCO EMS.</p>

```
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.
```


tibemsQueueSession

Type

- Purpose**Session restricted to queues.
- Remarks**Use this type for queue requestors.

Otherwise, when coding new programs, use the more general type, `tibemsSession` on page 306. Nonetheless, for backward compatibility, this type also supports existing programs that use it (rather than generic sessions).

Function	Description	Page
Messages		
<code>tibemsQueueSession_CreateBrowser</code>	Create a queue browser.	336
<code>tibemsQueueSession_CreateReceiver</code>	Create a queue receiver.	337
<code>tibemsQueueSession_CreateSender</code>	Create a message producer.	338
<code>tibemsQueueSession_CreateTemporaryQueue</code>	Create a temporary queue.	339
<code>tibemsQueueSession_DeleteTemporaryQueue</code>	Delete a temporary queue.	340

- Related Types**`tibemsSession` on page 306
- See Also**`tibemsQueueRequestor_Create` on page 209

tibemsQueueSession_CreateBrowser

Function

Purpose	Create a queue browser.
C Declaration	<pre>tibems_status tibemsQueueSession_CreateBrowser(tibemsQueueSession session, tibemsQueueBrowser* browser, tibemsQueue queue, const char* messageSelector);</pre>
COBOL Call	<pre>CALL "tibemsQueueSession_CreateBrowser" USING BY VALUE session, BY REFERENCE browser, BY VALUE queue, BY REFERENCE messageSelector, RETURNING tibems-status END-CALL.</pre>



session, browser and queue have usage pointer.

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 Message Selectors on page 21. When null, or the empty string, the browser views all messages in the queue.

See Also tibemsQueue on page 150
 tibemsQueueBrowser on page 352

tibemsQueueSession_CreateReceiver

Function

Purpose Create a queue receiver.

C Declaration

```
tibems_status tibemsQueueSession_CreateReceiver(
    tibemsQueueSession session,
    tibemsQueueReceiver* queueReceiver,
    tibemsQueue queue,
    const char* messageSelector );
```

COBOL Call

```
CALL "tibemsQueueSession_CreateReceiver"
    USING BY VALUE session,
          BY REFERENCE queueReceiver,
          BY VALUE queue,
          BY REFERENCE messageSelector,
          RETURNING tibems-status
END-CALL.
```



session, queueReceiver and queue have usage pointer.

Parameter	Description
session	Create the receiver in this session.
queueReceiver	The function stores the new receiver object in this location.
queue	Create a receiver for this queue.
messageSelector	When non-null, the server filters messages using this selector, so the receiver receives only matching messages; see Message Selectors on page 21. When null, or the empty string, the receiver receives messages without filtering.

See Also tibemsQueue on page 150
tibemsQueueReceiver on page 171

tibemsQueueSession_CreateSender

Function

- Purpose

Create a message producer.
- C Declaration

```
tibems_status tibemsQueueSession_CreateSender(  
    tibemsQueueSession session,  
    tibemsQueueSender* queueSender,  
    tibemsQueue queue );
```
- COBOL Call

```
CALL "tibemsQueueSession_CreateSender"  
    USING BY VALUE session,  
          BY REFERENCE queueSender,  
          BY VALUE queue,  
          RETURNING tibems-status  
END-CALL.
```



session, queueSender and queue have usage pointer.

Parameter	Description
session	Create the sender in this session.
queueSender	The function stores the new sender object in this location.
queue	When non-null, the sender sends messages to this queue. When null, the client program must specify the destination queue for each message individually.

See Also

tibemsQueue on page 150
tibemsQueueSender on page 194

tibemsQueueSession_CreateTemporaryQueue

Function

Purpose

Create a temporary queue.
This call is identical to `tibemsSession_CreateTemporaryQueue` on page 322.

C Declaration

```
tibems_status tibemsQueueSession_CreateTemporaryQueue(  
    tibemsQueueSession session,  
    tibemsTemporaryQueue* tmpQueue );
```

COBOL Call

```
CALL "tibemsQueueSession_CreateTemporaryQueue"  
    USING BY VALUE session,  
          BY REFERENCE tmpQueue,  
          RETURNING tibems-status  
END-CALL.
```



session and tmpQueue have usage pointer.

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.

See Also

`tibemsTemporaryQueue` on page 154

tibemsQueueSession_DeleteTemporaryQueue

Function

Purpose

Delete a temporary queue.
This call is identical to tibemsSession_DeleteTemporaryQueue on page 325.

C Declaration

```
tibems_status tibemsQueueSession_DeleteTemporaryQueue(  
    tibemsQueueSession session,  
    tibemsTemporaryQueue tmpQueue );
```

COBOL Call

```
CALL "tibemsQueueSession_DeleteTemporaryQueue"  
    USING BY VALUE session,  
          BY VALUE tmpQueue,  
          RETURNING tibems-status  
END-CALL.
```



session and tmpQueue have usage pointer.

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.

See Also

tibemsTemporaryQueue on page 154

tibemsTopicSession

Type

- Purpose

Session restricted to topics.
- Remarks

Use this type for topic requestors.

Otherwise, when coding new programs, use the more general type, `tibemsSession` on page 306. Nonetheless, for backward compatibility, this type also supports existing programs that use it (rather than generic sessions).

Function	Description	Page
Messages		
<code>tibemsTopicSession_CreateDurableSubscriber</code>	Create a durable topic subscriber.	342
<code>tibemsTopicSession_CreateSubscriber</code>	Create a topic subscriber.	345
<code>tibemsTopicSession_CreatePublisher</code>	Create a publisher.	344
<code>tibemsTopicSession_CreateTemporaryTopic</code>	Create a temporary topic.	347
<code>tibemsTopicSession_DeleteTemporaryTopic</code>	Delete a temporary topic.	348
<code>tibemsTopicSession_Unsubscribe</code>	Delete a subscription.	349

- Related Types

`tibemsSession` on page 306
- See Also

`tibemsTopicRequestor_Create` on page 211

tibemsTopicSession_CreateDurableSubscriber

Function

Purpose Create a durable topic subscriber.

C Declaration

```
tibems_status tibemsTopicSession_CreateDurableSubscriber(  
    tibemsTopicSession session,  
    tibemsTopicSubscriber* topicSubscriber,  
    tibemsTopic topic,  
    const char* subscriberName,  
    const char* messageSelector,  
    tibems_bool noLocal );
```

COBOL Call

```
CALL "tibemsTopicSession_CreateDurableSubscriber"  
    USING BY VALUE session,  
          BY REFERENCE topicSubscriber,  
          BY VALUE topic,  
          BY REFERENCE subscriberName,  
          BY REFERENCE messageSelector,  
          BY VALUE noLocal,  
          RETURNING tibems-status  
END-CALL.
```



session, topicSubscriber and topic have usage pointer.

Parameter	Description
session	Create the subscriber in this session.
topicSubscriber	The function stores the new subscriber object in this location.
topic	Create a subscriber for this topic.
subscriberName	Create a subscription in the server with this name.
messageSelector	<p>When non-null, the server filters messages using this selector, so the subscriber receives only matching messages; see Message Selectors on page 21.</p> <p>When null, or the empty string, the subscriber receives messages without filtering.</p>

Parameter	Description
<code>noLocal</code>	<p>When <code>true</code>, 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 <code>false</code>, the subscriber receives messages with local origin.</p>
See Also	<p><code>tibemsTopic</code> on page 156</p> <p><code>tibemsTopicSubscriber</code> on page 173</p>

tibemsTopicSession_CreatePublisher

Function

Purpose	Create a publisher.
C Declaration	<pre>tibems_status tibemsTopicSession_CreatePublisher(tibemsTopicSession session, tibemsTopicPublisher* topicPublisher, tibemsTopic topic);</pre>
COBOL Call	<pre>CALL "tibemsTopicSession_CreatePublisher" USING BY VALUE session, BY REFERENCE topicPublisher, BY VALUE topic, RETURNING tibems-status END-CALL.</pre>



session, topicPublisher and topic have usage pointer.

Parameter	Description
session	Create the publisher in this session.
topicPublisher	The function stores the new publisher object in this location.
topic	When non-null, the sender sends messages to this topic. When null, the client program must specify the destination topic for each message individually.

See Also tibemsTopic on page 156
 tibemsTopicPublisher on page 199

tibemsTopicSession_CreateSubscriber

Function

Purpose Create a topic subscriber.

C Declaration `tibems_status tibemsTopicSession_CreateSubscriber(
tibemsTopicSession session,
tibemsTopicSubscriber* topicSubscriber,
tibemsTopic topic,
const char* messageSelector,
tibems_bool noLocal);`

COBOL Call `CALL "tibemsTopicSession_CreateSubscriber"
USING BY VALUE session,
BY REFERENCE topicSubscriber,
BY VALUE topic,
BY REFERENCE messageSelector,
BY VALUE noLocal,
RETURNING tibems-status
END-CALL.`



session and topicSubscriber have usage pointer.

Parameter	Description
session	Create the subscriber in this session.
topicSubscriber	The function stores the new subscriber object in this location.
topic	Create a subscriber for this topic.
messageSelector	When non-null, the server filters messages using this selector, so the subscriber receives only matching messages; see Message Selectors on page 21. When null, or the empty string, the subscriber receives messages without filtering.
noLocal	When true, the server filters messages so the subscriber does not receive messages that originate locally—that is, messages sent through the same connection. When false, the subscriber receives messages with local origin.

See Also tibemsTopic on page 156

`tibemsTopicSubscriber` on page 173

tibemsTopicSession_CreateTemporaryTopic

Function

Purpose

Create a temporary topic.
This call is identical to `tibemsSession_CreateTemporaryTopic` on page 323.

C Declaration

```
tibems_status tibemsTopicSession_CreateTemporaryTopic(  
    tibemsTopicSession session,  
    tibemsTemporaryTopic* tmpTopic );
```

COBOL Call

```
CALL "tibemsTopicSession_CreateTemporaryTopic"  
    USING BY VALUE session,  
          BY REFERENCE tmpTopic,  
          RETURNING tibems-status  
END-CALL.
```



session and tmpTopic have usage pointer.

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 topics associated with the connection.

See Also

`tibemsTemporaryTopic` on page 155

tibemsTopicSession_DeleteTemporaryTopic

Function

Purpose Delete a temporary topic.
This call is identical to tibemsSession_DeleteTemporaryTopic on page 326.

C Declaration `tibems_status tibemsTopicSession_DeleteTemporaryTopic(
tibemsTopicSession session,
tibemsTemporaryTopic tmpTopic);`

COBOL Call `CALL "tibemsTopicSession_DeleteTemporaryTopic"
USING BY VALUE session,
BY VALUE tmpTopic,
RETURNING tibems-status
END-CALL.`



session and tmpTopic have usage pointer.

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.

See Also tibemsTemporaryTopic on page 155

tibemsTopicSession_Unsubscribe

Function

Purpose

Delete a subscription.

This call is identical to `tibemsSession_Unsubscribe` on page 332.

C Declaration

```
tibems_status tibemsTopicSession_Unsubscribe(  
    tibemsTopicSession session,  
    const char* name);
```

COBOL Call

```
CALL "tibemsTopicSession_Unsubscribe"  
    USING BY VALUE session,  
          BY REFERENCE name,  
          RETURNING tibems-status  
END-CALL.
```



`session` have usage pointer.

Parameter	Description
<code>session</code>	Delete a subscription in this session.
<code>name</code>	Delete this subscriber from the server.

Remarks

This call 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` or `tibemsTopicSubscriber` exists.

It is illegal to delete a subscription while one of its messages is either unacknowledged, or uncommitted (in the current transaction).

See Also

`tibemsTemporaryQueue` on page 154

Chapter 11 Queue Browser

Queue browsers let client programs examine the messages on a queue without removing them from the queue.

Topics

- *tibemsQueueBrowser*, page 352

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` on page 311.

Function	Description	Page
<code>tibemsQueueBrowser_Close</code>	Close the browser; reclaim resources.	353
<code>tibemsQueueBrowser_GetNext</code>	Get the next message from the browser.	355
<code>tibemsQueueBrowser_GetMsgSelector</code>	Get the selector string for the browser.	354
<code>tibemsQueueBrowser_GetQueue</code>	Get the queue that the browser examines.	356

See Also `tibemsSession_CreateBrowser` on page 311

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.

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.

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 on page 21.

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.

Parameter	Description
queueBrowser	Get the next message from the queue of this browser.
msg	The function stores the next message in this location.

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.

Parameter	Description
queueBrowser	Get the queue of this browser.
queue	The function stores the queue in this location.

Chapter 12 **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).

Topics

- *tibemsLookupContext*, page 358
- *tibemsLookupParams*, page 365

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	Page
tibemsLookupContext_Create tibemsLookupContext_CreateSSL	Create a new EMS lookup context object.	359
tibemsLookupContext_CreateExternal	Create a new LDAP lookup context object.	361
tibemsLookupContext_Destroy	Destroy a lookup context and reclaim resources.	362
tibemsLookupContext_Lookup tibemsLookupContext_LookupDestination tibemsLookupContext_LookupConnectionFactory	Lookup an object in the naming server.	363

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.

Parameter	Description
context	The function stores the new lookup context object in this location.
brokerURL	The context object connects to the EMS server at this URL.
username	The context object identifies itself to the server with this user name.
password	The context object identifies itself to the server with this password.

Parameter	Description
sslParams	The context object uses this data to create an SSL connection to the EMS server.
pk_password	The context object uses this password to decrypt its SSL private key.
Remarks	<p>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.</p> <p>If the server permits anonymous lookup, you may supply null values for the username and password parameters.</p>
See Also	tibemsLookupContext_CreateExternal on page 361

tibemsLookupContext_CreateExternal

Function

Purpose	Create a new LDAP lookup context object.
C Declaration	<pre>tibems_status tibemsLookupContext_CreateExternal(tibemsLookupContext* context, tibemsLookupParams lookupParams);</pre>
COBOL Call	<pre>CALL "tibemsLookupContext_CreateExternal" USING BY REFERENCE context, BY VALUE lookupParams, RETURNING tibems-status END-CALL.</pre>
	context and lookupParams have usage pointer.

Parameter	Description
context	The function stores 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	<p>tibemsLookupContext_Create on page 359</p> <p>tibemsLookupParams on page 365</p> <p>tibemsLookupParams_Create on page 366</p> <p>tibemsLookupParams_Destroy on page 367</p>
----------	--

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.

Parameter	Description
context	Destroy this lookup context object.

tibemsLookupContext_Lookup

Function

Purpose Lookup 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.

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.

Parameter	Description
context	Destroy this lookup context object.
name	Lookup this name.
object destination factory	The function stores the results of the lookup operation in this location.

tibemsLookupParams

Type

Purpose Encapsulate parameters for LDAP lookup.

Function	Description	Page
<code>tibemsLookupParams_Create</code>	Create a new LDAP lookup context object.	366
<code>tibemsLookupParams_Destroy</code>	Destroy a lookup parameters object.	367
<code>tibemsLookupParams_GetLdapServerUrl</code>	Get the LDAP server URL.	368
<code>tibemsLookupParams_SetLdapBaseDN</code>	Set the LDAP base distinguished name.	369
<code>tibemsLookupParams_SetLdapCAFile</code>	Set the LDAP CA certificate file.	370
<code>tibemsLookupParams_SetLdapCAPath</code>	Set the LDAP CA certificate directory pathname.	371
<code>tibemsLookupParams_SetLdapCertFile</code>	Set the LDAP client certificate file.	372
<code>tibemsLookupParams_SetLdapCiphers</code>	Set SSL ciphers.	373
<code>tibemsLookupParams_SetLdapConnType</code>	Set the LDAP connection type.	374
<code>tibemsLookupParams_SetLdapCredential</code>	Set the LDAP credential (password).	375
<code>tibemsLookupParams_SetLdapKeyFile</code>	Set the LDAP client key file.	376
<code>tibemsLookupParams_SetLdapPrincipal</code>	Set the LDAP principal (user name).	377
<code>tibemsLookupParams_SetLdapRandFile</code>	Set a randomization data file.	378
<code>tibemsLookupParams_SetLdapSearchScope</code>	Set the LDAP search scope.	379
<code>tibemsLookupParams_SetLdapServerUrl</code>	Set the LDAP server URL.	380

tibemsLookupParams_Create

Function

Purpose Create a new lookup parameters object.

C Declaration `tibemsLookupParams
tibemsLookupParams_Create(void);`

COBOL Call `CALL "tibemsLookupParams_Create"
RETURNING lparams
END-CALL.`



`lparams` has usage pointer.

See Also `tibemsLookupContext_CreateExternal` on page 361

tibemsLookupParams_Destroy

Function

Purpose	Destroy a lookup parameters object.
C Declaration	<pre>void tibemsLookupParams_Destroy(tibemsLookupParams lparams);</pre>
COBOL Call	<pre>CALL "tibemsLookupParams_Destroy" USING BY VALUE lparams END-CALL.</pre>



lparams has usage pointer.

Parameter	Description
lparams	The function stores the new lookup context object in this location.

tibemsLookupParams_GetLdapServerUrl

Function

Purpose	Get the LDAP server URL.
C Declaration	<pre>char* tibemsLookupParams_GetLdapServerUrl(tibemsLookupParams lparams);</pre>
COBOL Call	<pre>CALL "tibemsLookupParams_GetLdapServerUrl" USING BY VALUE lparams END-CALL.</pre>



lparams has usage pointer.

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);`

COBOL Call `CALL "tibemsLookupParams_SetLdapBaseDN"
USING BY VALUE lparams,
BY REFERENCE basedn,
RETURNING tibems-status
END-CALL.`



lparams has usage pointer.

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 );
```
- COBOL Call

```
CALL "tibemsLookupParams_SetLdapCAFile"  
    USING BY VALUE lparams,  
          BY REFERENCE file,  
          RETURNING tibems-status  
END-CALL.
```



lparams has usage pointer.

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);`

COBOL Call `CALL "tibemsLookupParams_SetLdapCAPath"
USING BY VALUE lparams,
BY REFERENCE path,
RETURNING tibems-status
END-CALL.`



lparams has usage pointer.

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	<pre>tibems_status tibemsLookupParams_SetLdapCertFile(tibemsLookupParams lparams, const char* file);</pre>
COBOL Call	<pre>CALL "tibemsLookupParams_SetLdapCertFile" USING BY VALUE lparams, BY REFERENCE file, RETURNING tibems-status END-CALL.</pre>



lparams has usage pointer.

Parameter	Description
lparams	Set the property in this parameter object.
file	Set the property to this filename.

Remarks	<p>The client program reads the client certificate from this file.</p> <p>Set this parameter when the LDAP server requires clients to present certificates as identification.</p>
---------	---

tibemsLookupParams_SetLdapCiphers

Function

Purpose	Set SSL ciphers.
C Declaration	<pre>tibems_status tibemsLookupParams_SetLdapCiphers(tibemsLookupParams lparams, const char* ciphers);</pre>
COBOL Call	<pre>CALL "tibemsLookupParams_SetLdapCiphers" USING BY VALUE lparams, BY REFERENCE ciphers, RETURNING tibems-status END-CALL.</pre>



lparams has usage pointer.

Parameter	Description
lparams	Set the property in this parameter object.
ciphers	<p>Set the property to specify these ciphers (as a colon-separated list).</p> <p>For example:</p> <p>"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"</p>

Remarks	<p>Set this parameter when using secure LDAP connections.</p> <p>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.</p>
---------	--

tibemsLookupParams_SetLdapConnType

Function

Purpose	Set the LDAP connection type.
C Declaration	<pre>tibems_status tibemsLookupParams_SetLdapConnType(tibemsLookupParams lparams, const char* type);</pre>
COBOL Call	<pre>CALL "tibemsLookupParams_SetLdapConnType" USING BY VALUE lparams, BY REFERENCE type, RETURNING tibems-status END-CALL.</pre>



lparams has usage pointer.

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 );
```
- COBOL Call

```
CALL "tibemsLookupParams_SetLdapCredential"  
    USING BY VALUE lparams,  
          BY REFERENCE credential,  
          RETURNING tibems-status  
END-CALL.
```



lparams has usage pointer.

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 );
```
- COBOL Call

```
CALL "tibemsLookupParams_SetLdapKeyFile"  
    USING BY VALUE lparams,  
          BY REFERENCE file,  
          RETURNING tibems-status  
END-CALL.
```



lparams has usage pointer.

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 );
```
- COBOL Call

```
CALL "tibemsLookupParams_SetLdapPrincipal"  
    USING BY VALUE lparams,  
          BY REFERENCE principal,  
          RETURNING tibems-status  
END-CALL.
```



lparams has usage pointer.

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 );
```
- COBOL Call

```
CALL "tibemsLookupParams_SetLdapRandFile"  
    USING BY VALUE lparams,  
          BY REFERENCE file,  
          RETURNING tibems-status  
END-CALL.
```



lparams has usage pointer.

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);`

COBOL Call `CALL "tibemsLookupParams_SetLdapSearchScope"
USING BY VALUE lparams,
BY REFERENCE scope,
RETURNING tibems-status
END-CALL.`



lparams has usage pointer.

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	<pre>tibems_status tibemsLookupParams_SetLdapServerUrl(tibemsLookupParams lparams, const char* url);</pre>
COBOL Call	<pre>CALL "tibemsLookupParams_SetLdapServerUrl" USING BY VALUE lparams, BY REFERENCE url, RETURNING tibems-status END-CALL.</pre>



lparams has usage pointer.

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.

Chapter 13 **XA—External Transaction Manager**

This chapter describes the EMS XA API. This API lets you code your own transaction manager (TM) that can interact with the EMS server as a transactional resource.

Topics

- *tibemsXAConnection_Close*, page 383
- *tibemsXAConnection_Create*, page 385
- *tibemsXAConnection_CreateXASession*, page 387
- *tibemsXAConnection_Get*, page 388
- *tibemsXAConnection_GetXASession*, page 389
- *XID*, page 390
- *tibemsXAResource*, page 391
- *tibemsXASession*, page 404

tibemsXAConnection

This implicit type is not defined; instead it is identical to `tibemsConnection` on page 214. 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.

Parameter	Description
connection	Close this connection.

Remarks	<p>Closing an XA connection reclaims all XA resources associated with the connection or its sessions.</p> <p>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.</p> <p>Closing a connection deletes all temporary destinations associated with the connection.</p>
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 <i>not</i> force acknowledgment in client-acknowledged sessions. When the program still has a message that it received from a connection that has since closed, <code>tibemsMsg_Acknowledge</code> returns the status code <code>TIBEMS_ILLEGAL_STATE</code> .
Transactions	Closing a connection rolls back all open transactions in all sessions associated with the connection.

See Also `tibemsMsg_Acknowledge` on page 28
 `tibemsMsgConsumer` on page 162
 `tibemsMsgProducer` on page 178
 `tibemsDestination` on page 144
 `tibemsSession` on page 306

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.  
  
CALL "tibemsXAConnection_CreateSSL"  
    USING BY REFERENCE connection,  
          BY REFERENCE brokerURL,  
          BY REFERENCE clientId,  
          BY REFERENCE username,  
          BY REFERENCE password,  
          BY VALUE sslParams,  
          BY REFERENCE pk-password,  
          RETURNING tibems-status  
END-CALL.
```



connection and sslParams have usage pointer.

Parameter	Description
connection	The function stores the new connection in this location.
brokerURL	Find the EMS server at this URL.

Parameter	Description
clientId	Identify the client program to the server with this unique ID.
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
TIBEMS_OK	The call succeeded.
TIBEMS_SERVER_NOT_CONNECTED	<ul style="list-style-type: none">• No server is running at the specified URL.• The call could not communicate with a server because of mismatched SSL and TCP protocols.• Other error situations are possible.
TIBEMS_SECURITY_EXCEPTION	<ul style="list-style-type: none">• The server rejected the connection because the username or password was invalid.• SSL setup is incorrect.
TIBEMS_INVALID_CLIENT_ID	The client ID is not unique; that is, another client already uses the ID.

See Also `tibemsQueueConnection_Create` on page 230
 `tibemsTopicConnection_Create` on page 234

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.

Remarks The new session has transactional semantics with an external transaction manager, and uses the connection for all server communications.

Parameter	Description
connection	Create a session on this connection.
session	The function stores the new session in this location.



XA sessions do not support routed queues.

See Also tibemsSession on page 306

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.

Parameter	Description
connection	The function stores the connection object in this location.
brokerURL	Find the connection the EMS server at this URL.

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.

Parameter	Description
connection	Get the XA session object from this XA connection.
xaSession	The function 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	Page
<code>tibemsXAResource_Commit</code>	Commit a transaction.	392
<code>tibemsXAResource_End</code>	Disassociate a transaction from the resource.	393
<code>tibemsXAResource_Forget</code>	Unimplemented.	394
<code>tibemsXAResource_GetRMID</code>	Get the RMID of the resource.	395
<code>tibemsXAResource_GetTransactionTimeout</code>	Get the timeout limit.	396
<code>tibemsXAResource_isSameRM</code>	Determine whether two resource objects originate from the same connection to an EMS server.	397
<code>tibemsXAResource_Prepere</code>	Prepare a transaction for commit.	398
<code>tibemsXAResource_Recover</code>	Get a list of prepared transactions.	399
<code>tibemsXAResource_SetRMID</code>	Assign an RMID to a resource.	401
<code>tibemsXAResource_SetTransactionTimeout</code>	Set the timeout limit.	402
<code>tibemsXAResource_Start</code>	Associate a transaction with a resource.	403

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.

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.

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	<pre>tibems_status tibemsXAResource_Forget(tibemsXAResource xaResource, XID* xid);</pre>
COBOL Call	<pre>CALL "tibemsXAResource_Forget" USING BY VALUE xaResource, BY REFERENCE xid, RETURNING tibems-status END-CALL.</pre>



xaResource and xid have usage pointer.

Parameter	Description
xaResource	Forget the transaction in this resource.
xid	Forget this transaction.

Remarks	<p>In the XA interface, the transaction manager can instruct a resource to forget a transaction.</p> <p>However, this call is not implemented in EMS; it is present for completeness only.</p>
---------	--

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.

Parameter	Description
xaResource	Get the RMID of this resource.
rmid	The function stores 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.

Parameter	Description
xaResource	Get the transaction timeout of this resource.
seconds	The function stores the transaction timeout (in seconds) in this location.

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 on page 402

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.

Parameter	Description
xaResource	First XA resource.
xaResource2	Second XA resource.
result	The function stores the result in this location:

tibemsXAResource_Prepere

Function

- Purpose

Prepare a transaction for commit.
- C Declaration

```
tibems_status tibemsXAResource_Prepere(  
    tibemsXAResource xaResource,  
    XID* xid );
```
- COBOL Call

```
CALL "tibemsXAResource_Prepere"  
    USING BY VALUE xaResource,  
          BY REFERENCE xid,  
          RETURNING tibems-status  
END-CALL.
```



xaResource and xid have usage pointer.

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	<pre>tibems_status tibemsXAResource_Recover(tibemsXAResource xaResource, XID* xids, tibems_int desiredCount, tibems_int* returnedCount, tibems_int flag);</pre>
COBOL Call	<pre>CALL "tibemsXAResource_Recover" USING BY VALUE xaResource, BY REFERENCE xids, BY VALUE desiredCount, BY REFERENCE returnedCount, BY VALUE flag, RETURNING tibems-status END-CALL.</pre>



xaResource and xids have usage pointer.

Parameter	Description
xaResource	List the prepared transactions of this resource.
xids	The function stores the list of transaction IDs in the array at this location.
desiredCount	Size of the array (number of XIDs).
returnedCount	The function stores the actual number of transaction IDs in this location.
flag	TMSTARTRSCAN—Start a new list of XIDs; the EMS server generates a complete list, and sends the first batch. TMNOFLAGS—Continue the list of XIDs; the EMS server sends the next batch. TMENDRSCAN—The EMS server discards its list of prepared transactions, and reclaims storage.

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	<pre>tibems_status tibemsXAResource_Rollback(tibemsXAResource xaResource, XID* xid);</pre>
COBOL Call	<pre>CALL "tibemsXAResource_Rollback" USING BY VALUE xaResource, BY REFERENCE xid, RETURNING tibems-status END-CALL.</pre>



xaResource and xid have usage pointer.

Parameter	Description
xaResource	Roll back in this resource.
xid	Roll back this transaction.

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.

Parameter	Description
xaResource	Set the RMID of this resource.
rmid	Use this RMID.

tibemsXAResource_SetTransactionTimeout

Function

Purpose	Set the timeout limit.
C Declaration	<pre>tibems_status tibemsXAResource_SetTransactionTimeout(tibemsXAResource xaResource, tibems_int seconds);</pre>
COBOL Call	<pre>CALL "tibemsXAResource_SetTransactionTimeout" USING BY VALUE xaResource, BY VALUE seconds, RETURNING tibems-status END-CALL.</pre>



xaResource has usage pointer.

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 on page 396

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.

Parameter	Description
xaResource	Associate a transaction with this resource.
xid	Associate this transaction with a resource.
flags	TMNOFLAGS—This is a new transaction, which was not previously associated with the resource. TMRESUME—This transaction was previously associated with the resource.

tibemsXASession

Type

Purpose A session with an XA resource.

Function	Description	Page
tibemsXASession_Close	Close an XA session.	405
tibemsXASession_GetSession	Get the EMS session from an XA session.	406
tibemsXASession_GetXAResource	Get the XA resource of an XA session.	407



XA sessions do not support routed queues.

See Also tibemsSession on page 306

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.

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.

Parameter	Description
xaSession	Get the EMS session from this XA session.
session	The function stores the EMS session in this location.

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.

Parameter	Description
session	Get the XA resource from this session.
xaResource	The function stores the resource in this location.

Chapter 14 **Types**

This chapter presents types in general use.

Topics

- *Uniform Types, page 410*

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;
```

Table 17 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

Chapter 15 Utilities

This chapter presents utility functions.

Topics

- *tibems_GetConnectAttemptCount*, page 414
- *tibems_GetConnectAttemptDelay*, page 415
- *tibems_GetReconnectAttemptCount*, page 416
- *tibems_GetReconnectAttemptDelay*, page 417
- *tibems_GetSocketReceiveBufferSize*, page 418
- *tibems_GetSocketSendBufferSize*, page 419
- *tibems_SetConnectAttemptCount*, page 420
- *tibems_SetConnectAttemptDelay*, page 421
- *tibems_SetReconnectAttemptCount*, page 422
- *tibems_SetReconnectAttemptDelay*, page 423
- *tibems_SetSocketReceiveBufferSize*, page 424
- *tibems_SetSocketSendBufferSize*, page 425
- *tibems_Sleep*, page 426
- *tibems_Version*, page 427

Utility Functions

The following functions are not related to a specific data type.

tibems_Close

Function

Purpose	Cleanup before unloading the EMS client library.
C Declaration	<code>void tibems_close();</code>
Remarks	Windows platforms allow selective unloading of DLLs. Before unloading the EMS library, you must call this function to clean up.

tibems_GetConnectAttemptCount

Function

Purpose	Return the connection attempts limit.
C Declaration	<code>tibems_int tibems_GetConnectAttemptCount(void);</code>
COBOL Call	<code>CALL "tibems_GetConnectAttemptCount" RETURNING tibems-Return-Value END-CALL.</code>
Remarks	This setting governs all client <code>tibemsConnection</code> 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	<code>tibems_SetConnectAttemptCount</code> on page 420

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` on page 421

tibems_GetReconnectAttemptCount

Function

Purpose	Return the reconnection attempts limit.
C Declaration	<code>tibems_int tibems_GetReconnectAttemptCount(void);</code>
COBOL Call	<code>CALL "tibems_GetReconnectAttemptCount" RETURNING tibems-Return-Value END-CALL.</code>
Remarks	This setting governs all client <code>tibemsConnection</code> 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	<code>tibems_GetConnectAttemptCount</code> on page 414 <code>tibems_SetReconnectAttemptCount</code> on page 422

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` on page 415
`tibems_SetReconnectAttemptDelay` on page 423

tibems_GetSocketReceiveBufferSize

Function

Purpose	Return the size of socket receive buffers.
C Declaration	<code>tibems_int tibems_GetSocketReceiveBufferSize(void);</code>
COBOL Call	<code>CALL "tibems_GetSocketReceiveBufferSize" RETURNING tibems-Return-Value END-CALL.</code>
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	<code>tibems_SetSocketReceiveBufferSize</code> on page 424

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` on page 425

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.

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` on page 414

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.

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` on page 415

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.

Parameter	Description
count	Set the reconnect limit to this value. The default value is 4. The minimum value is 1.

See Also `tibems_GetReconnectAttemptCount` on page 416

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.

Parameter	Description
delay	Set the reconnect delay to this value. The default value is 500. The minimum value is 250.

See Also `tibems_GetReconnectAttemptDelay` on page 417

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.

Parameter	Description
size	Sockets use receive buffers of this size (in kilobytes).

See Also tibems_GetSocketReceiveBufferSize on page 418

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.

Parameter	Description
size	Sockets use send buffers of this size (in kilobytes).

See Also tibems_GetSocketSendBufferSize on page 419

tibems_Sleep

Function

Purpose	Sleep thread.
C Declaration	<pre>void tibems_Sleep(tibems_long milliseconds);</pre>
COBOL Call	<pre>CALL "tibems_Sleep" USING BY VALUE milliseconds END-CALL.</pre>



COBOL usage of milliseconds is COMP-2.

Parameter	Description
milliseconds	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*).

Chapter 16 **Exception**

This chapter presents exceptions related to EMS.

Topics

- *tibems_status*, page 430
- *tibemsStatus_GetText*, page 440

tibems_status

Type

Purpose Functions return status codes to indicate return conditions.

Table 18 Status Codes (Sheet 1 of 6)

Constant	Code	Description
TIBEMS_OK	0	The call completed normally.
TIBEMS_ILLEGAL_STATE	1	<p>A function call or server request occurred in an inappropriate context.</p> <p>For example, <code>tibemsSession_Commit</code> indicates this status when the session is non-transactional.</p>
TIBEMS_INVALID_CLIENT_ID	2	<p>The provider rejects the connection’s client ID.</p> <p>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.)</p>
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 Message Selectors on page 21.
TIBEMS_EXCEPTION	5	Non-specific error code.
TIBEMS_SECURITY_EXCEPTION	6	<p>The function cannot complete because of a security restriction.</p> <p>For example, the provider rejects a user or the user’s authentication.</p>

Table 18 Status Codes (Sheet 2 of 6)

Constant	Code	Description
TIBEMS_MSG_EOF	7	<p>The data stream within a message ended unexpectedly.</p> <p><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.</p>
TIBEMS_MSG_NOT_READABLE	9	Attempt to read from a message in write-only mode.
TIBEMS_MSG_NOT_WRITEABLE	10	<p>Attempt to write to a message in read-only mode.</p> <p>See also, <code>tibemsMsg_MakeWriteable</code> on page 55.</p>
TIBEMS_SERVER_NOT_CONNECTED	11	<ul style="list-style-type: none"> • An attempt to connect to the server has failed. • The operation requires a server connection, but the program is not connected.
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	<p>The connection URL includes an invalid hostname, or an attempt to lookup the host address failed.</p> <p>Host names must be less than 128 characters.</p>
TIBEMS_INVALID_PORT	18	The connection URL includes an invalid port number.

Table 18 Status Codes (Sheet 3 of 6)

Constant	Code	Description
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 <code>tibemsExceptionCallback</code> on page 237
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, <code>tibemsLookupContext</code> on page 358.
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, <code>tibemsLookupContext</code> on page 358 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.

Table 18 Status Codes (Sheet 2 of 6)

Constant	Code	Description
TIBEMS_MSG_EOF	7	<p>The data stream within a message ended unexpectedly.</p> <p><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.</p>
TIBEMS_MSG_NOT_READABLE	9	Attempt to read from a message in write-only mode.
TIBEMS_MSG_NOT_WRITEABLE	10	<p>Attempt to write to a message in read-only mode.</p> <p>See also, <code>tibemsMsg_MakeWriteable</code> on page 55.</p>
TIBEMS_SERVER_NOT_CONNECTED	11	<ul style="list-style-type: none"> • An attempt to connect to the server has failed. • The operation requires a server connection, but the program is not connected.
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	<p>The connection URL includes an invalid hostname, or an attempt to lookup the host address failed.</p> <p>Host names must be less than 128 characters.</p>
TIBEMS_INVALID_PORT	18	The connection URL includes an invalid port number.

Table 18 Status Codes (Sheet 3 of 6)

Constant	Code	Description
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 <code>tibemsExceptionCallback</code> on page 237
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, <code>tibemsLookupContext</code> on page 358.
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, <code>tibemsLookupContext</code> on page 358 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.

Table 18 Status Codes (Sheet 2 of 6)

Constant	Code	Description
TIBEMS_MSG_EOF	7	<p>The data stream within a message ended unexpectedly.</p> <p><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.</p>
TIBEMS_MSG_NOT_READABLE	9	Attempt to read from a message in write-only mode.
TIBEMS_MSG_NOT_WRITEABLE	10	<p>Attempt to write to a message in read-only mode.</p> <p>See also, <code>tibemsMsg_MakeWriteable</code> on page 55.</p>
TIBEMS_SERVER_NOT_CONNECTED	11	<ul style="list-style-type: none"> • An attempt to connect to the server has failed. • The operation requires a server connection, but the program is not connected.
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	<p>The connection URL includes an invalid hostname, or an attempt to lookup the host address failed.</p> <p>Host names must be less than 128 characters.</p>
TIBEMS_INVALID_PORT	18	The connection URL includes an invalid port number.

Table 18 Status Codes (Sheet 3 of 6)

Constant	Code	Description
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 <code>tibemsExceptionCallback</code> on page 237
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, <code>tibemsLookupContext</code> on page 358.
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, <code>tibemsLookupContext</code> on page 358 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.

Table 18 Status Codes (Sheet 4 of 6)

Constant	Code	Description
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 <code>tibemsMsgConsumer_ReceiveTimeout</code> on page 168.
TIBEMS_INTR	51	A blocking operation has been interrupted. See <code>tibemsMsgConsumer_Receive</code> on page 166.
TIBEMS_QUEUE_LIMIT_EXCEEDED	52	A server queue 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	Mainframe only. A blocking operation has been interrupted. See <code>tibx_MVSConsole_SetConsumer()</code> on page 445.
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.

Table 18 Status Codes (Sheet 5 of 6)

Constant	Code	Description
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.
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.
SSL		
TIBEMS_INVALID_CERT	150	SSL detected an invalid X.509 certificate.

Table 18 Status Codes (Sheet 6 of 6)

Constant	Code	Description
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.
TIBEMS_INVALID_CRL_DATA	159	SSL detected a corrupt certificate revocation list (CRL) file; it could not load the CRL file.
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.`

Parameter	Description
status	Get the text corresponding to this status code.

Chapter 17 **IBM Mainframe**

This chapter presents items of interest only to programmers coding for IBM mainframe computers.

Topics

- *tibems_SetCodePages()*, page 443
- *tibx_MVSConsole_SetConsumer()*, page 445
- *tibx_MVSConsole_Create()*, page 446

IBM Mainframe Calls

The calls we present on the following pages are implemented only on IBM mainframe platforms.

tibems_SetCodePages()

Function

C Declaration	<pre>tibems_status tibems_SetCodePages(char* host_codepage, char* net_codepage);</pre>
COBOL Call	<pre>CALL "tibems_SetCodePages" USING BY REFERENCE host-codepage, BY REFERENCE net-codepage, RETURNING TIBEMS-STATUS END-CALL.</pre>
Purpose	Set code pages for string conversion on EBCDIC platforms (when non-default code pages are required).
String Conversion	<p>EMS software uses the operating system's <code>iconv()</code> call to automatically convert strings within messages. Conversion occurs only as needed:</p> <ul style="list-style-type: none">• Programs running in EBCDIC environments represent all strings using an EBCDIC code page (called the <i>host code page</i>). Before sending a message, the EMS client library converts its strings to an ASCII or UTF-8 character set (the <i>network code page</i>).• 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	<p>This call sets the host and network code pages for string conversions in EBCDIC environments.</p> <p>Call this function when the system code pages differ from the EMS default code pages (see the table of Default Code Pages on page 444). Throughout an enterprise, all sending and receiving programs must use the same code pages.</p> <p>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.</p> <p>Programs may call this function at most once. The call <i>must</i> precede the first call to any message function, and the arrival of the first message from the network.</p>

(Sheet 1 of 2)

Parameter	Description
host_codepage	Set this code page as the native (EBCDIC) character encoding for the host computer.

(Sheet 2 of 2)

Parameter	Description
net_codepage	Set this code page as the ASCII or UTF-8 character set for the network.

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
"IBM-1047"	"ISO8859-1"

See Also

Strings and Character Encodings on page 4

tibx_MVSConsole_SetConsumer()

Function

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.

Purpose Exit from a blocking listener.

Remarks Programs in single-threaded environments (such as COBOL) need a way to interrupt blocking receive calls (such as tibemsMsgConsumer_Receive).
This call is available only in MVS.
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).

Parameter	Description
pConsole	Set the consumer of this MVS console object.
tibemsMsgConsumer	Use this message consumer.
pFunction	In COBOL, use this tibems_MVS_BreakFunction function address.
tibems_MVS_BreakFunction	In C, use this break function name.

See Also tibemsMsgConsumer_Receive on page 166
tibx_MVSConsole_Create() on page 446

tibx_MVSConsole_Create()

Function

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.

Purpose Create or destroy an MVS console.

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() on page 445).

This call is available only in MVS.

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.

(Sheet 1 of 2)

Parameter	Description
pConsole	The create call stores the MVS console handle in this location. The destroy call destroys this MVS console.

(Sheet 2 of 2)

Parameter	Description
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.

See Also tibemsMsgConsumer_Receive on page 166
 tibx_MVSConsole_SetConsumer() on page 445

Console_Response

Function Type

- C Declaration

```
signed long int
Console_Response(
    signed long int rc,
    char* ops_command );
```
- Purpose

Callback function to relay the results of MVS console commands to C programs.
- Remarks

Available only in C in MVS.
COBOL does not support callback functions.

Parameter	Description
rc	This parameter receives the return status code from the MVS console.
ops_command	This parameter receives the operator command from the MVS console.
return value	Zero indicates a normal return. All other values indicate a special console command—namely, stop or shut.

See Also tibx_MVSConsole_Create() on page 446

Index

A

Acknowledge 28
assembly 5

B

body type 138
body types, message 13
BytesMessage
 Write methods 87

C

cache, assembly 5
character encoding 4
checklist, programmer's 5
clear body 30
clear properties 31
code pages 443
compression 3
Connection 214
ConnectionConsumer (not supported) 3
console, MVS 446
Console_Response 448
conversion, data type 24
copy a message 33
create a message 32
create message from byte array 34
CreateQueueSession 232
customer support xviii

D

data type conversion 24
delivery mode 130
Destination
 overview 140
destroy message 35
dynamic destination 140

E

EBCDIC 443
encoding, character 4
exceptions 429

F

fault tolerance
 ActiveURL 221

G

Get
 MapMessage 93
get
 message properties 57
global assembly cache 5

H

headers, message 14

L

- library files 6
- linking 6
- LookupContext 358

M

- MapMessage
 - Get methods 93
 - set methods 99
- Message
 - Acknowledge 28
 - body types 13
 - headers 14
 - parts of 12
 - properties 18
 - set property methods 60
- message
 - get property 57
 - selectors 21
- message body type 138
- MessageConsumer 162
- MessageProducer.Close 180
- MessageProducer.Send 186
- MVS console 446

O

- ObjectMessage 106

P

- property, message 18
 - set 60
- proxy, SSL 291, 293
- Publish 201
- publish 201

Q

- Queue 150
- QueueBrowser 352
- QueueConnection
 - CreateQueueSession 232

R

- Read
 - StreamMethod 114
- read
 - tibemsBytesMsg 80
- read-only 30, 34
- request 14

S

- selectors, message 21
- ServerSession (not supported) 3
- ServerSessionPool (not supported) 3
- Set
 - MapMessage 99
 - message property 60
- set
 - code pages 443
- SSL 3
- SSL proxy 291, 293
- static destination 140
- StreamMessage
 - Read methods 114
 - Write methods 120
- string and character encoding 4
- string encoding
 - EBCDIC 443
- support, contacting xviii

T

- technical support xviii
- temporary destination 140
- tibems_Close 413
- tibems_GetConnectAttemptCount 414
- tibems_GetConnectAttemptDelay 415
- tibems_GetReconnectAttemptCount 416
- tibems_GetReconnectAttemptDelay 417
- tibems_GetSocketReceiveBufferSize 418
- tibems_GetSocketSendBufferSize 419
- tibems_MVS_BreakFunction 445
- tibems_SetCodePages() 443
- tibems_SetConnectAttemptCount 420
- tibems_SetConnectAttemptDelay 421
- tibems_SetReconnectAttemptCount 422
- tibems_SetReconnectAttemptDelay 423
- tibems_SetSocketReceiveBufferSize 424
- tibems_SetSocketSendBufferSize 425
- tibems_Sleep 426
- tibems_status 430
- tibems_Version 427
- tibemsAcknowledgeMode 333
- tibemsBytesMsg 76
 - read 80
- tibemsBytesMsg_Create 77
- tibemsBytesMsg_GetBodyLength 78
- tibemsBytesMsg_GetBytes 79
- tibemsBytesMsg_ReadBoolean 80
- tibemsBytesMsg_ReadByte 80
- tibemsBytesMsg_ReadBytes 84
- tibemsBytesMsg_ReadChar 80
- tibemsBytesMsg_ReadDouble 80
- tibemsBytesMsg_ReadFloat 80
- tibemsBytesMsg_ReadInt 80
- tibemsBytesMsg_ReadLong 80
- tibemsBytesMsg_ReadShort 80
- tibemsBytesMsg_ReadUnsignedByte 80
- tibemsBytesMsg_ReadUnsignedShort 80
- tibemsBytesMsg_ReadUTF 80
- tibemsBytesMsg_Reset 85
- tibemsBytesMsg_SetBytes 86
- tibemsBytesMsg_WriteBoolean 87
- tibemsBytesMsg_WriteByte 87
- tibemsBytesMsg_WriteBytes 90
- tibemsBytesMsg_WriteChar 87
- tibemsBytesMsg_WriteDouble 87
- tibemsBytesMsg_WriteFloat 87
- tibemsBytesMsg_WriteInt 87
- tibemsBytesMsg_WriteLong 87
- tibemsBytesMsg_WriteShort 87
- tibemsBytesMsg_WriteUTF 87
- tibemsConnection_Close 216
- tibemsConnection_Create 218
- tibemsConnection_CreateSession 220
- tibemsConnection_CreateSSL 218
- tibemsConnection_GetActiveURL 221
- tibemsConnection_GetClientId 222
- tibemsConnection_GetExceptionListener 223
- tibemsConnection_SetClientId 224
- tibemsConnection_SetExceptionListener 225
- tibemsConnection_Start 226
- tibemsConnection_Stop 227
- tibemsConnectionFactory 270
- tibemsConnectionFactory_Create 273
- tibemsConnectionFactory_CreateConnection 274
- tibemsConnectionFactory_CreateConnectionSSL 275
- tibemsConnectionFactory_Destroy 277
- tibemsConnectionFactory_GetType 282
- tibemsConnectionFactory_SetClientId 283
- tibemsConnectionFactory_SetConnectAttemptCount 284
- tibemsConnectionFactory_SetConnectAttemptDelay 285
- tibemsConnectionFactory_SetMetric 286
- tibemsConnectionFactory_SetReconnectAttemptCount 287
- tibemsConnectionFactory_SetReconnectAttemptDelay 288
- tibemsConnectionFactory_SetServerURL 289
- tibemsConnectionFactory_SetSSLParams 290
- tibemsConnectionFactory_SetSSLProxy 291
- tibemsConnectionFactory_SetSSLProxyAuth 293
- tibemsConnectionFactory_SetSSLProxyHost 278
- tibemsConnectionFactory_SetSSLProxyPassword 281
- tibemsConnectionFactory_SetSSLProxyPort 279
- tibemsConnectionFactory_SetSSLProxyUser 280
- tibemsConnectionFactory_SetType 294
- tibemsConnectionFactoryType 295
- tibemsd 5, 270, 357

tibemsDeliveryMode 130
 tibemsDestination 144
 tibemsDestination_Copy 145
 tibemsDestination_Create 146
 tibemsDestination_Destroy 147
 tibemsDestination_GetName 148
 tibemsDestination_GetType 149
 tibemsExceptionCallback 237
 tibemsFactoryLoadBalanceMetric 296
 tibemsLookupContext_Create 359
 tibemsLookupContext_CreateExternal 361
 tibemsLookupContext_CreateSSL 359
 tibemsLookupContext_Destroy 362
 tibemsLookupContext_Lookup 363
 tibemsLookupContext_LookupConnectionFactory 36
 3
 tibemsLookupContext_LookupDestination 363
 tibemsLookupParams 365
 tibemsLookupParams_Create 366
 tibemsLookupParams_Destroy 367
 tibemsLookupParams_GetLdapServerUrl 368
 tibemsLookupParams_SetLdapBaseDN 369
 tibemsLookupParams_SetLdapCAFile 370
 tibemsLookupParams_SetLdapCAPath 371
 tibemsLookupParams_SetLdapCertFile 372
 tibemsLookupParams_SetLdapCiphers 373
 tibemsLookupParams_SetLdapConnType 374
 tibemsLookupParams_SetLdapCredential 375
 tibemsLookupParams_SetLdapKeyFile 376
 tibemsLookupParams_SetLdapPrincipal 377
 tibemsLookupParams_SetLdapRandFile 378
 tibemsLookupParams_SetLdapSearchScope 379
 tibemsLookupParams_SetLdapServerUrl 380
 tibemsMapMsg 91
 tibemsMapMsg_Create 92
 tibemsMapMsg_GetBoolean 93
 tibemsMapMsg_GetByte 93
 tibemsMapMsg_GetBytes 93
 tibemsMapMsg_GetChar 93
 tibemsMapMsg_GetDouble 93
 tibemsMapMsg_GetField 93
 tibemsMapMsg_GetFloat 93
 tibemsMapMsg_GetInt 93
 tibemsMapMsg_GetLong 93
 tibemsMapMsg_GetMapMsg 93
 tibemsMapMsg_GetMapNames 97
 tibemsMapMsg_GetShort 94
 tibemsMapMsg_GetString 94
 tibemsMapMsg_ItemExists 98
 tibemsMapMsg_SetBoolean 99
 tibemsMapMsg_SetByte 99
 tibemsMapMsg_SetBytes 104
 tibemsMapMsg_SetChar 99
 tibemsMapMsg_SetDouble 99
 tibemsMapMsg_SetDoubleArray 99
 tibemsMapMsg_SetFloat 99
 tibemsMapMsg_SetFloatArray 100
 tibemsMapMsg_SetInt 99
 tibemsMapMsg_SetIntArray 100
 tibemsMapMsg_SetLong 99
 tibemsMapMsg_SetLongArray 100
 tibemsMapMsg_SetMapMsg 100
 tibemsMapMsg_SetReferencedBytes 104
 tibemsMapMsg_SetShort 99
 tibemsMapMsg_SetShortArray 100
 tibemsMapMsg_SetStreamMsg 100
 tibemsMapMsg_SetString 99
 tibemsMsg 25
 tibemsMsg_Acknowledge 28
 tibemsMsg_ClearBody 30
 tibemsMsg_ClearProperties 31
 tibemsMsg_Create 32
 tibemsMsg_CreateCopy 33
 tibemsMsg_CreateFromBytes 34
 tibemsMsg_Destroy 35
 tibemsMsg_GetAsBytes 36
 tibemsMsg_GetAsBytesCopy 38
 tibemsMsg_GetBodyType 40
 tibemsMsg_GetBooleanProperty 57
 tibemsMsg_GetByteProperty 57
 tibemsMsg_GetByteSize 41
 tibemsMsg_GetCorrelationID 42
 tibemsMsg_GetDeliveryMode 44
 tibemsMsg_GetDestination 45
 tibemsMsg_GetDoubleProperty 57
 tibemsMsg_GetEncoding 46
 tibemsMsg_GetExpiration 47
 tibemsMsg_GetFloatProperty 57
 tibemsMsg_GetIntProperty 57
 tibemsMsg_GetLongProperty 57

tibemsMsg_GetMessageID 48
 tibemsMsg_GetPriority 49
 tibemsMsg_GetProperty 57
 tibemsMsg_GetPropertyNames 50
 tibemsMsg_GetRedelivered 51
 tibemsMsg_GetReplyTo 52
 tibemsMsg_GetShortProperty 57
 tibemsMsg_GetStringProperty 57
 tibemsMsg_GetTimestamp 53
 tibemsMsg_GetType 54
 tibemsMsg_MakeWriteable 55
 tibemsMsg_Print 56
 tibemsMsg_PrintFile 56
 tibemsMsg_PropertyExists 63
 tibemsMsg_SetBooleanProperty 60
 tibemsMsg_SetByteProperty 60
 tibemsMsg_SetCorrelationID 64
 tibemsMsg_SetDeliveryMode 66
 tibemsMsg_SetDestination 67
 tibemsMsg_SetDoubleProperty 60
 tibemsMsg_SetEncoding 68
 tibemsMsg_SetExpiration 69
 tibemsMsg_SetFloatProperty 60
 tibemsMsg_SetIntProperty 60
 tibemsMsg_SetLongProperty 60
 tibemsMsg_SetMessageID 70
 tibemsMsg_SetPriority 71
 tibemsMsg_SetRedelivered 72
 tibemsMsg_SetReplyTo 73
 tibemsMsg_SetShortProperty 60
 tibemsMsg_SetStringProperty 60
 tibemsMsg_SetTimestamp 74
 tibemsMsg_SetType 75
 tibemsMsg_Callback 176
 tibemsMsgConsumer_Close 163
 tibemsMsgConsumer_GetMsgListener 164
 tibemsMsgConsumer_GetMsgSelector 165
 tibemsMsgConsumer_Receive 166
 tibemsMsgConsumer_ReceiveNoWait 167
 tibemsMsgConsumer_ReceiveTimeout 168
 tibemsMsgConsumer_SetMsgListener 170
 tibemsMsgEnum_Destroy 132
 tibemsMsgEnum_GetNextName 133
 tibemsMsgField_Print 137
 tibemsMsgField_PrintFile 137
 tibemsMsgProducer 178
 tibemsMsgProducer_Close 180
 tibemsMsgProducer_GetDeliveryMode 181
 tibemsMsgProducer_GetDisableMessageID 182
 tibemsMsgProducer_GetDisableMessageTimestamp 183
 tibemsMsgProducer_GetPriority 184
 tibemsMsgProducer_GetTimeToLive 185
 tibemsMsgProducer_Send 186
 tibemsMsgProducer_SendEx 186
 tibemsMsgProducer_SendToDestination 186
 tibemsMsgProducer_SendToDestinationEx 186
 tibemsMsgProducer_SetDeliveryMode 189
 tibemsMsgProducer_SetDisableMessageID 190
 tibemsMsgProducer_SetDisableMessageTimestamp 191
 tibemsMsgProducer_SetPriority 192
 tibemsMsgProducer_SetTimeToLive 193
 tibemsMsgRequestor_Close 207
 tibemsMsgRequestor_Request 208
 tibemsMsgType 138
 tibemsObjectMsg_Create 107
 tibemsObjectMsg_GetObjectBytes 108
 tibemsObjectMsg_SetObjectBytes 109
 tibemsQueue_Create 151
 tibemsQueue_Destroy 152
 tibemsQueue_GetQueueName 153
 tibemsQueueBrowser_Close 353
 tibemsQueueBrowser_GetMsgSelector 354
 tibemsQueueBrowser_GetNext 355
 tibemsQueueBrowser_GetQueue 356
 tibemsQueueConnection 229
 tibemsQueueConnection_Create 230
 tibemsQueueConnection_CreateSSL 230
 tibemsQueueConnectionFactory 297
 tibemsQueueConnectionFactory_CreateConnection 298
 tibemsQueueConnectionFactory_CreateConnectionSSL 299
 tibemsQueueReceiver 171
 tibemsQueueReceiver_GetQueue 172
 tibemsQueueRequestor 206
 tibemsQueueRequestor_Create 209
 tibemsQueueSender 194
 tibemsQueueSender_GetQueue 195

tibemsQueueSender_Send 196
 tibemsQueueSender_SendEx 196
 tibemsQueueSender_SendToQueue 196
 tibemsQueueSender_SendToQueueEx 196
 tibemsQueueSession 335
 tibemsQueueSession_CreateBrowser 336
 tibemsQueueSession_CreateReceiver 337
 tibemsQueueSession_CreateSender 338
 tibemsQueueSession_CreateTemporaryQueue 339
 tibemsQueueSession_DeleteTemporaryQueue 340
 tibemsSession 306
 tibemsSession_Close 309
 tibemsSession_Commit 310
 tibemsSession_CreateBrowser 311
 tibemsSession_CreateBytesMessage 312
 tibemsSession_CreateConsumer 313
 tibemsSession_CreateDurableSubscriber 315
 tibemsSession_CreateMapMessage 318
 tibemsSession_CreateMessage 319
 tibemsSession_CreateProducer 320
 tibemsSession_CreateStreamMessage 321
 tibemsSession_CreateTemporaryQueue 322
 tibemsSession_CreateTemporaryTopic 323
 tibemsSession_CreateTextMessage 324
 tibemsSession_CreateTextMessageEx 324
 tibemsSession_DeleteTemporaryQueue 325
 tibemsSession_DeleteTemporaryTopic 326
 tibemsSession_GetAcknowledgeMode 327
 tibemsSession_GetTransacted 328
 tibemsSession_Recover 329
 tibemsSession_Rollbackm 331
 tibemsSession_Unsubscribe 332
 tibemsSSL_GetDebugTrace 239
 tibemsSSL_GetTrace 239
 tibemsSSL_OpenSSLVersion 240
 tibemsSSL_SetDebugTrace 241
 tibemsSSL_SetTrace 241
 tibemsSSLHostNameVerifier 266
 tibemsSSLParams_AddIssuerCert 244
 tibemsSSLParams_AddIssuerCertFile 244
 tibemsSSLParams_AddTrustedCert 246
 tibemsSSLParams_AddTrustedCertFile 246
 tibemsSSLParams_Create 248
 tibemsSSLParams_Destroy 249
 tibemsSSLParams_GetIdentity 250
 tibemsSSLParams_GetPrivateKey 251
 tibemsSSLParams_SetAuthOnly 252
 tibemsSSLParams_SetCiphers 253
 tibemsSSLParams_SetCRLPath 254
 tibemsSSLParams_SetCRLUpdateInterval 255
 tibemsSSLParams_SetExpectedHostName 256
 tibemsSSLParams_SetHostNameVerifier 257
 tibemsSSLParams_SetIdentity 258
 tibemsSSLParams_SetIdentityFile 258
 tibemsSSLParams_SetPrivateKey 259
 tibemsSSLParams_SetPrivateKeyFile 259
 tibemsSSLParams_SetRandData 260
 tibemsSSLParams_SetRandEGD 260
 tibemsSSLParams_SetRandFile 260
 tibemsSSLParams_SetRenegotiateInterval 262
 tibemsSSLParams_SetRenegotiateSize 263
 tibemsSSLParams_SetVerifyHost 264
 tibemsSSLParams_SetVerifyHostName 264
 tibemsStatus_GetText 440
 tibemsStreamMsg 110
 tibemsStreamMsg_Create 112
 tibemsStreamMsg_FreeField 113
 tibemsStreamMsg_ReadBoolean 114
 tibemsStreamMsg_ReadByte 114
 tibemsStreamMsg_ReadBytes 117
 tibemsStreamMsg_ReadChar 114
 tibemsStreamMsg_ReadDouble 114
 tibemsStreamMsg_ReadField 118
 tibemsStreamMsg_ReadFloat 114
 tibemsStreamMsg_ReadInt 114
 tibemsStreamMsg_ReadLong 114
 tibemsStreamMsg_ReadShort 114
 tibemsStreamMsg_ReadString 114
 tibemsStreamMsg_Reset 119
 tibemsStreamMsg_WriteBoolean 120
 tibemsStreamMsg_WriteByte 120
 tibemsStreamMsg_WriteBytes 124
 tibemsStreamMsg_WriteChar 120
 tibemsStreamMsg_WriteDouble 120
 tibemsStreamMsg_WriteDoubleArray 120
 tibemsStreamMsg_WriteFloat 120
 tibemsStreamMsg_WriteFloatArray 120
 tibemsStreamMsg_WriteInt 120
 tibemsStreamMsg_WriteIntArray 120
 tibemsStreamMsg_WriteLong 120

tibemsStreamMsg_WriteLongArray 121
 tibemsStreamMsg_WriteMapMsg 121
 tibemsStreamMsg_WriteShort 120
 tibemsStreamMsg_WriteShortArray 121
 tibemsStreamMsg_WriteStreamMsg 121
 tibemsStreamMsg_WriteString 120
 tibemsTemporaryQueue 154
 tibemsTemporaryTopic 155
 tibemsTextMsg 125
 tibemsTextMsg_Create 126
 tibemsTextMsg_GetText 127
 tibemsTextMsg_SetText 128
 tibemsTopic_Create 157
 tibemsTopic_Destroy 158
 tibemsTopic_GetTopicName 159
 tibemsTopicConnection 233
 tibemsTopicConnection_Create 234
 tibemsTopicConnection_CreateSSL 234
 tibemsTopicConnection_CreateTopicSession 236
 tibemsTopicConnectionFactory 301
 tibemsTopicConnectionFactory_CreateConnectio 302
 tibemsTopicConnectionFactory_CreateConnectionSSL 303
 tibemsTopicPublisher_GetTopic 200
 tibemsTopicPublisher_Publish 201
 tibemsTopicPublisher_PublishEx 201
 tibemsTopicPublisher_PublishToTopic 201
 tibemsTopicPublisher_PublishToTopicEx 201
 tibemsTopicRequestor 206
 tibemsTopicRequestor_Create 211
 tibemsTopicSession 341
 tibemsTopicSession_CreateDurableSubscriber 342
 tibemsTopicSession_CreatePublisher 344
 tibemsTopicSession_CreateSubscriber 345
 tibemsTopicSession_CreateTemporaryTopic 347
 tibemsTopicSession_DeleteTemporaryTopic 348
 tibemsTopicSession_Unsubscribe 349
 tibemsTopicSubscriber 173
 tibemsTopicSubscriber_GetNoLocal 174
 tibemsTopicSubscriber_GetTopic 175
 tibemsXAConnection_Close 383
 tibemsXAConnection_Create 385
 tibemsXAConnection_CreateSSL 385
 tibemsXAConnection_CreateXASession 387
 tibemsXAConnection_Get 388
 tibemsXAConnection_GetXASession 389
 tibemsXAResource 391
 tibemsXAResource_Commit 392
 tibemsXAResource_End 393
 tibemsXAResource_Forget 394
 tibemsXAResource_GetRMID 395
 tibemsXAResource_GetTransactionTimeout 396
 tibemsXAResource_isSameRM 397
 tibemsXAResource_Prepare 398
 tibemsXAResource_Recover 399
 tibemsXAResource_Rollback 400
 tibemsXAResource_SetRMID 401
 tibemsXAResource_SetTransactionTimeout 402
 tibemsXAResource_Start 403
 tibemsXASession_Close 405
 tibemsXASession_GetSession 406
 tibemsXASession_GetXAResource 407
 tibx_MVSCConsole_Create 446
 tibx_MVSCConsole_Destroy 446
 tibx_MVSCConsole_SetConsumer 445
 Topic 156
 TopicPublisher 199
 Publish 201
 TopicPublisher.Publish 201
 translation, character encoding 4
 type conversion 24

U

Unicode 4

W

Write

 BytesMessage 87

 StreamMessage 120

X

- XA (not supported) 3
- XID 390