

TIBCO Enterprise Message Service™

User's Guide

*Software 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 ActiveEnterprise, TIBCO Adapter, TIBCO Hawk, TIBCO Rendezvous, TIBCO Enterprise, TIBCO Enterprise Message Service, and the TIBCO logo 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

Figures	xiii
Tables	xv
Preface	xvii
Related Documentation	xviii
TIBCO Enterprise Message Service Documentation	xviii
Other TIBCO Product Documentation	xviii
Third Party Documentation	xviii
How to Contact TIBCO Customer Support	xx
Chapter 1 Overview	1
JMS Overview	2
JMS Message Models	3
Point-to-Point	3
Publish and Subscribe	4
Bridges Between Destinations	5
Controlling the Flow of Messages	5
Performance Features of Queues	5
Additional Queue and Topic Features	6
Client APIs	7
TIBCO Rendezvous Java Applications	7
Messages	8
String Encoding	9
Message Tracing and Monitoring	9
Sample Code	10
Message Priority	11
Administration	12
Administering the Server	12
User and Group Management	13
Using TIBCO Hawk	13
Fault Tolerance	14
Routing	15
SSL	16

Integrating With Third-Party Products	17
Transaction Support.	17
Chapter 2 Working With the Client API	19
API Overview	20
C and C# Client APIs.	20
Programming Model.	20
ConnectionFactory	25
Connection	26
Session	27
MessageProducer.	28
MessageConsumer.	29
Durable Subscribers for Topics	29
Synchronous or Asynchronous Message Consumption	29
MessageListener.	30
Chapter 3 Working With Destinations	31
Destination Overview	32
Static Queues and Topics	32
Dynamic Queues and Topics	32
Temporary Queues and Topics	33
Destination Bridges	33
Destination Properties.	34
Wildcards	44
Wildcards * and >.	44
Wildcards in Topics	44
Wildcards in Queues	45
Inheritance	46
Inheritance of Properties	46
Inheritance of Permissions.	46
Destination Bridges.	48
Creating a Bridge	49
Access Control and Bridges.	50
Transactions	50
Flow Control	51
Enabling Flow Control	51
Enforcing Flow Control.	51
Routes and Flow Control	52
Destination Bridges and Flow Control	52
Flow Control, Threads and Deadlock.	53

Chapter 4 Working With Messages	55
JMS Message Structure	56
Header Fields	56
Properties	57
Message Bodies	57
Maximum Message Size	58
Message Persistence	59
File Locking	59
Character Encoding in Messages	60
Supported Character Encodings	61
Sending Messages	61
Receiving Messages	63
Message Compression	65
About Message Compression	65
Setting Message Compression	65
Message Acknowledgement	66
Undelivered Message Queue	67
Including the Message Sender	68
Message Extensions	69
EMS Message Delivery Mode Extensions	70
Reliable Message Delivery	70
No-Acknowledgement Message Receipt	71
 Chapter 5 Working With TIBCO Rendezvous	 73
Overview	74
Message Translation	74
Configuration	74
Deprecated Configuration Properties	75
Configuring Transports for Rendezvous	76
Transport Definitions	76
Topics	79
Import Only when Subscribers Exist	79
Wildcards	80
Certified Messages	80
Queues	81
Configuration	81
Import—Start and Stop	81
Wildcards	81
Import Issues	83
Import Destination Names Must be Unique	83
JMSReplyTo	83

Guaranteed Delivery	83
Export Issues	84
JMSReplyTo.	84
Certified Messages	84
Guaranteed Delivery	84
Message Translation.	85
JMS Header Fields	85
JMS Property Fields.	86
Message Body.	87
Data Types.	88
Pure Java Rendezvous Programs	91
Chapter 6 Working With TIBCO SmartSockets	95
Overview	96
Message Translation	96
Configuration	96
Starting the Servers	97
Configuring Transports for SmartSockets	98
Transport Definitions	98
Subscribe Mode.	102
Destination Name—Syntax and Semantics.	102
Topics	104
Import Only when Subscribers Exist	104
Wildcards	105
Queues	106
Configuration	106
Import—Start and Stop	106
Wildcards	106
Import Issues.	108
Import Destination Names Must be Unique	108
JMSReplyTo.	108
Guaranteed Delivery	108
Export Issues	109
JMSReplyTo.	109
Wildcard Subscriptions	109
Guaranteed Delivery	109
Message Translation.	111
JMS Header Fields	111
JMS Property Fields.	111
SmartSockets Message Properties	112
Message Body.	113
Data Types.	115

Destination Names	116
Chapter 7 Using the Configuration Files	117
Location of Configuration Files	118
Mechanics of Configuration	119
Using the Main Configuration File	120
Using Other Configuration Files	148
users	148
groups	149
topics	150
queues	150
acl	151
bridges	152
routes	152
factories	154
transports	156
tibrvcm	157
durables	157
Chapter 8 Using the Administration Tool	161
Starting the Administration Tool	162
When You First Start tibemsadmin	164
Naming Conventions	166
Command Listing	167
Chapter 9 Authentication and Permissions.	199
Overview of Users, Groups, and Permissions	200
Setting Up Access Control	202
Enabling Access Control	203
Server Control	203
Destination Control	204
Users and Groups	205
Users	205
Groups	205
Configuring an External Directory	206
Setting Permissions	210
Example of Setting Permissions	211
Inheritance of Permissions	211
Revoking Permissions	213
When Permissions Are Checked	214
Example of Permission Checking	214

Administrator Permissions	216
Predefined Administrative User and Group	216
Granting and Revoking Administration Permissions	217
Enforcement of Administrator Permissions	218
Global Administrator Permissions	218
Destination-Level Permissions.	221
Protection Permissions	223
Chapter 10 Monitoring Server Activity	225
Log Files and Tracing	226
Configuring the Log File.	226
Tracing on the Server	227
Message Tracing.	231
Enabling Message Tracing for a Destination	231
Enabling Message Tracing on a Message.	232
Monitoring Server Events	233
System Monitor Topics.	233
Monitoring Messages.	233
Viewing Monitor Topics	236
Performance Implications of Monitor Topics	237
Working with Server Statistics.	238
Overall Server Statistics.	238
Enabling Statistic Gathering.	239
Displaying Statistics.	241
Chapter 11 Deploying the Application	243
Running the Server.	244
Starting the Server.	244
emsntsreg	246
Security Considerations	248
Secure Environment	248
Destination Security	248
Authorization Parameter	248
Admin Password	249
Connection Security.	249
Communication Security	249
Sources of Authentication Data	250
Timestamp	250
Passwords	250
Audit Trace Logs	251
Running TIBCO Enterprise Message Service Client-Side Application	252
Programmer's Checklist.	252

Connecting Directly to TIBCO Enterprise Message Service Server	253
Using JNDI with TIBCO Enterprise Message Service	255
Dynamic Topics and Queues	255
Static Topics and Queues	255
Example	256
Using SSL with JNDI Lookups	257
Performing Fault-Tolerant JNDI Lookups	263
Chapter 12 Using the SSL Protocol	265
SSL Support in TIBCO Enterprise Message Service	266
Digital Certificates	267
Digital Certificate File Formats	267
Private Key Formats	268
Overview of the SSL Protocol	269
Cipher Suite Negotiation	270
Client and Server Authentication	271
Renegotiating the Session Key	274
File Names for Certificates and Keys	275
Configuring SSL in the Server	276
SSL Parameters	276
Command Line Options	276
Configuring SSL in EMS Clients	277
Client Digital Certificates	277
Configuring SSL	278
Specifying Cipher Suites	281
Syntax for Cipher Suites	281
Supported Cipher Suites	283
SSL Authentication Only	286
Third-Party SSL Hardware Accelerators	287
Ingrian Accelerator	287
Chapter 13 Fault Tolerance	289
Fault Tolerance Overview	290
Failover	292
Detection	292
Response	292
Message Redelivery	293
Heartbeat Parameters	294
Shared State	295
Implementing Shared State	295
Messages Stored in Shared State	297

Storage Files	297
Storage Parameters	298
Configuring Fault-Tolerant Servers	299
Authorization and Fault-Tolerant Servers	299
SSL	299
Reconnect Timeout	300
Configuring Clients for Fault-Tolerant Connections.	301
Specifying More Than Two URLs	302
Chapter 14 Working With Routes	303
Overview of Routing	304
Route	305
Basic Operation	305
Global Destinations	306
Unique Routing Path	306
Zone	308
Basic Operation	308
Eliminating Redundant Paths with a One-Hop Zone	308
Overlapping Zones	309
Active and Passive Routes	311
Configuring Routes and Zones	312
Routes to Fault-Tolerant Servers	313
Routing and SSL	313
Routed Topic Messages	317
Propagating Registered Interest	317
Selectors for Routing Topic Messages.	319
Routed Queues	322
Routing and Authorization.	325
Appendix A Using the Samples	327
Starting Work with the Client Sample Files	328
Compiling the Sample Files	328
Publish and Subscribe Example	329
Overview of the Example	329
Creating a Topic	329
Creating Users	331
Publishing and Subscribing	332
Running Client Samples	332
Appendix B Using TIBCO Hawk	335
Overview of Server Management With TIBCO Hawk	336

- Installing the Classes 337
 - Windows Installation 337
 - UNIX Installation 338
 - Parameters 339
- Method Description 341
- Appendix C Monitor Messages 345**
 - Description of Monitor Topics 346
 - Description of Topic Message Properties 349
- Appendix D Error and Status Messages 359**
 - Error and Status Messages 360
- Index 395**

Figures

Figure 1	Message Delivery	3
Figure 2	Point-to-point messages	4
Figure 3	Publish and subscribe messages	5
Figure 4	JMS API programming model	21
Figure 5	Specific interfaces for topics and queues	22
Figure 6	Bridging a topic to a queue	48
Figure 7	Bridging a topic to multiple destinations	49
Figure 8	Flow Control Deadlock across Two Threads	53
Figure 9	Clients sending UTF-8 encoded messages	62
Figure 10	Clients sending messages with a specific encoding	63
Figure 11	Clients receiving messages	64
Figure 12	Message Delivery and Acknowledgement	66
Figure 13	Rendezvous Transports in the EMS Server	74
Figure 14	SmartSockets Transports in the EMS Server	96
Figure 15	Users, groups, and permissions	201
Figure 16	SSL Protocol	269
Figure 17	Ingrian Accelerator	287
Figure 18	Primary Server and Backup Server	290
Figure 19	Routes: bidirectionality and corresponding destinations	305
Figure 20	Routes: global destinations	306
Figure 21	Routes: Unique Path	307
Figure 22	Zones: multi-hop	308
Figure 23	Zones: one-hop	309
Figure 24	Zones: overlap	310
Figure 25	Routing: Propagating Subscribers	317
Figure 26	Routing: Topic Selectors, example	319
Figure 27	Routing: Queues	322
Figure 28	Routing: Authorization	325

Tables

Table 1	Summary of message properties	8
Table 2	Summary of administration features.	12
Table 3	API object summary	22
Table 4	Destination properties	34
Table 5	Prefetch	41
Table 6	JMS Message Headers	56
Table 7	JMS Message Types	57
Table 8	Rendezvous: Transport Parameters	76
Table 9	Rendezvous: Mapping JMS Header Fields to RV Datatypes.	85
Table 10	Rendezvous: Mapping Message Types (Import)	87
Table 11	Rendezvous: Mapping Message Types (Export)	88
Table 12	Rendezvous: Mapping Data Types	88
Table 13	TibrvJMSTransport class	92
Table 14	SmartSockets: Transport Parameters	98
Table 15	SmartSockets Mapping Message Properties (Import & Export).	112
Table 16	SmartSockets: Mapping Message Types (Export).	114
Table 17	SmartSockets: Mapping Data Types	115
Table 18	Configuration parameters.	120
Table 19	tibemsadmin Options	163
Table 20	Set server parameters	178
Table 21	show connections: type Parameter	185
Table 22	show connections Table Information	185
Table 23	show durable Table Information.	187
Table 24	show durables Table Information	189
Table 25	show queue Table Information.	190
Table 26	show queues Table Information.	191
Table 27	show routes Table Information.	192
Table 28	show topic Table Information	194

Table 29	Show topics table information	196
Table 30	Default configuration for popular LDAP servers	208
Table 31	Queue Permission	210
Table 32	Topic Permission	210
Table 33	Global administrator permissions	219
Table 34	Destination-level administration permissions	222
Table 35	Server tracing options	228
Table 36	Message monitoring qualifiers	234
Table 37	tibemsd Options	244
Table 38	SSL properties for client applications using JNDI	257
Table 39	File types	275
Table 40	SSL JAR Files	277
Table 41	ConnectionFactory SSL parameters	279
Table 42	Qualifiers for Cipher Suites in Java Clients	281
Table 43	OpenSSL Qualifiers for Cipher Suites	282
Table 44	Supported Cipher Suites in Java API	283
Table 45	Shared Storage Criteria for Fault Tolerance	295
Table 46	Shared State Files	297
Table 47	SSL Parameters for Routes	314
Table 48	TIBCO Enterprise Message Service classes in TIBCO Hawk	336
Table 49	TIBCO Hawk MicroAgent Parameters	339
Table 50	TIBCO Hawk method names	341
Table 51	Monitor topics	346
Table 52	Message properties	349
Table 53	Event Action Property Values	355
Table 54	Event Reason Property Values	357

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 xviii*
- *How to Contact TIBCO Customer Support, page xx*

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* The C API reference is available in HTML and PDF formats.
- *TIBCO Enterprise Message Service Java API Reference* The Java API reference is available as JavaDoc, and you can access the reference only through the HTML documentation interface.
- *TIBCO Enterprise Message Service .NET API Reference* The .NET API 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 Overview

This chapter contains a general overview of Java Message Service (JMS) and TIBCO Enterprise Message Service™ (EMS) concepts.

Topics

- *JMS Overview, page 2*
- *JMS Message Models, page 3*
- *Client APIs, page 7*
- *Messages, page 8*
- *Sample Code, page 10*
- *Administration, page 12*
- *Fault Tolerance, page 14*
- *Routing, page 15*
- *SSL, page 16*
- *Integrating With Third-Party Products, page 17*

JMS Overview

Java Message Service 1.1 (JMS) is a Java framework specification for messaging between applications. Sun Microsystems developed this specification, in conjunction with TIBCO and others, to supply a uniform messaging interface among enterprise applications.

Using a message service allows you to integrate the applications within an enterprise. For example, you may have several applications: one for customer relations, one for product inventory, and another for raw materials tracking. Each application is crucial to the operation of the enterprise, but even more crucial is communication between the applications to ensure the smooth flow of business processes. Message-oriented-middleware (MOM) creates a common communication protocol between these applications and allows you to easily integrate new and existing applications in your enterprise computing environment.

The JMS framework (an interface specification, not an implementation) is designed to supply a basis for MOM development. TIBCO Enterprise Message Service implements JMS, and integrates support for connecting other message services (such as TIBCO Rendezvous and TIBCO SmartSockets). This chapter describes the concepts of JMS and its implementation in TIBCO Enterprise Message Service. For more information on JMS requirements and features, we recommend the following sources:

- Java Message Service specification, available through <http://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.

JMS Compliance

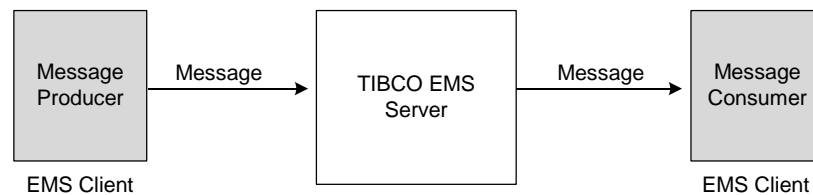
TIBCO Enterprise Message Service 4.3 has passed Sun Microsystems' Technology Compatibility Kit (TCK) for Java Message Service 1.1 (JMS 1.1), indicate that EMS 4.3 is compliant with the JMS 1.1 specification.

JMS Message Models

JMS is based on creation and delivery of messages. Messages are structured data that one application sends to another. The creator of the message is known as the *producer* and the receiver of the message is known as the *consumer*. The TIBCO EMS server acts as an intermediary for the message and sends it to the correct destination. The server also provides enterprise-class functionality such as fault-tolerance, message routing, and communication with other messaging systems, such as TIBCO Rendezvous™ and TIBCO SmartSockets™.

Figure 1 illustrates an application producing a message, sending it by way of the server, and a different application receiving the message.

Figure 1 Message Delivery



JMS supports two messaging models:

- Point-to-point (queues)
- Publish and subscribe (topics)

Point-to-Point

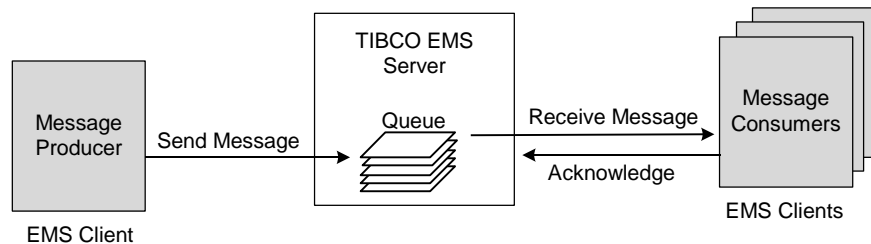
Point-to-point messaging has one producer and one consumer per message. This style of messaging uses a *queue* to store messages until they are received. The message producer sends the message to the queue; the message consumer retrieves messages from the queue and sends acknowledgement that the message was received.

More than one producer can send messages to the same queue, and more than one consumer can retrieve messages from the same queue. The queue can be configured to be exclusive, if desired. If the queue is exclusive, then all queue messages can only be retrieved by the first consumer specified for the queue. Exclusive queues are useful when you want only one application to receive messages for a specific queue. If the queue is not exclusive, any number of

receivers can retrieve messages from the queue. Non-exclusive queues are useful for balancing the load of incoming messages across multiple receivers. Regardless of whether the queue is exclusive or not, only one consumer can ever retrieve each message that is placed on the queue.

Figure 2 illustrates point-to-point messaging using a non-exclusive queue. Each message consumer receives a message from the queue and acknowledges receipt of the message. The message is taken off the queue so that no other consumer can receive it.

Figure 2 Point-to-point messages



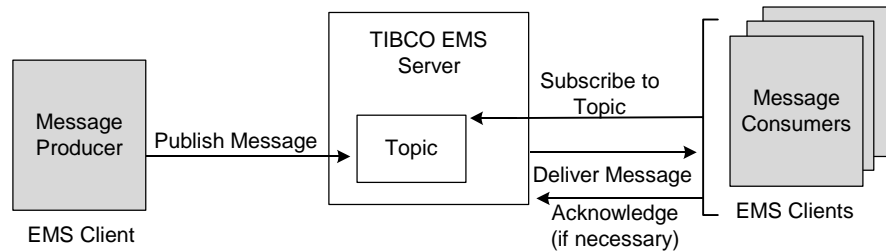
Publish and Subscribe

In a publish and subscribe message system, producers address messages to a topic. In this model, the producer is known as a *publisher* and the consumer is known as a *subscriber*.

Many publishers can publish to the same topic, and a message from a single publisher can be received by many subscribers. Subscribers subscribe to topics, and all messages published to the topic are received by all subscribers to the topic. This type of message protocol is also known as *broadcast* messaging because messages are sent over the network and received by all interested subscribers, similar to how radio or television signals are broadcast and received.

Figure 3 illustrates publish and subscribe messaging. Each message consumer subscribes to a topic. When a message is published to that topic, all subscribed consumers receive the message.

Figure 3 Publish and subscribe messages



There can be a time dependency in the publish and subscribe model. By default, subscribers only receive messages when they are active. If messages are delivered when the subscriber is not available, the subscriber does not receive those messages. JMS specifies a way to remove part of the timing dependency by allowing subscribers to create durable subscriptions. Messages for durable subscriptions are stored on the server until the message expires or the storage limit is reached. Subscribers can receive messages from a durable subscription even if the subscriber was not available when the message was originally delivered.

Bridges Between Destinations

You can also create bridges between destinations of the same or different types to create a hybrid messaging model for your application. This can be useful if your application requires that you send the same message to both a topic and a queue. For more information on creating bridges between destinations and situations where this may be useful, see [Destination Bridges](#) on page 48.

Controlling the Flow of Messages

You can control the flow of messages to a destination. This is useful when message producers send messages much faster than message consumers can receive them. For more information on flow control, see [Flow Control](#) on page 51.

Performance Features of Queues

You can specify that a queue receiver can receive a batch of messages in the background to improve performance. Alternatively, you can specify that queue receivers should only receive one message at a time, if you must guarantee that no queue receivers should receive more than one message at a time.

Additional Queue and Topic Features

TIBCO Enterprise Message Service allows you to specify several properties for topics and queues that enhance the functionality of each messaging model. Queue and topic properties are set when the destination is created. Queue and topic properties can add the following functionality:

- A fail safe mode allows messages to be written to disk synchronously to guarantee no messages are ever lost due to server failure.
- Enforcement of permissions can be set at the queue or topic level so that some destinations may require access control and others may not.
- You can limit the size of messages stored on a queue. If a receiver is offline for a long time, queue messages can accumulate and consume machine resources.
- You can limit the size of messages stored for durable topic subscriptions. If a subscriber is offline for a long time, topic messages can accumulate and consume machine resources.
- Messages sent to destinations can be routed to other servers.
- You can exchange messages with other message services. Queues can receive TIBCO Rendezvous and TIBCO SmartSockets messages. Topics can either receive or send Rendezvous and TIBCO SmartSockets messages.
- Queues can be set to be exclusive or non-exclusive. Only one receiver can receive messages from an exclusive queue. More than one receiver can receive messages from non-exclusive queues.
- Queues can specify a redelivery policy. When messages must be redelivered, you can specify a property on the queue that determines the maximum number of times a message should be redelivered.
- All messages passing through a destination can be traced and logged.
- The user name of message producer that sends messages can be included in the message.
- TIBCO Enterprise Message Service allows you to create wildcard destinations. The wildcard destination name is the parent, and any names that match the wildcard destination name inherit the properties of the parent.

See Chapter 3, *Working With Destinations*, on page 31 for more information about working with destinations.

Client APIs

Java applications use the `javax.jms` package to send or receive JMS messages. This is a standard set of interfaces, specified by the JMS specification, for creating the connection to the EMS server, specifying the type of message to send, and creating the destination (topic or queue) to send to or receive from. You can find a description of the `javax.jms` package in *TIBCO Enterprise Message Service Java API Reference* included in the online documentation. Because TIBCO Enterprise Message Service implements the JMS standard, you can also view the documentation on these interfaces along with the JMS specification at java.sun.com/products/jms/index.html.

TIBCO Enterprise Message Service includes a parallel API for C client programs; see *TIBCO Enterprise Message Service C & COBOL API Reference* (online documentation).

TIBCO Enterprise Message Service includes a parallel API for .NET client programs; see *TIBCO Enterprise Message Service .NET API Reference*.

TIBCO Rendezvous Java Applications

TIBCO Enterprise Message Service includes a Java class that allows pure Java TIBCO Rendezvous applications to connect directly with the TIBCO Enterprise Message Service server; see *Pure Java Rendezvous Programs* on page 91.

Messages

JMS messages have a standard structure. This structure includes the following sections:

- Header (required)
- Properties (optional)
- Body (optional)

The JMS specification details a standard format for the header and body of a message. Properties are provider-specific and can include information on specific implementations or enhancements to JMS functionality. Table 1 describes the message properties available with TIBCO Enterprise Message Service.

Table 1 Summary of message properties (Sheet 1 of 2)

Property	Description	More Info
JMS_TIBCO_COMPRESS	Allows messages to be compressed for more efficient storage.	65
JMS_TIBCO_DISABLE_SENDER	Specifies that the user name of the message sender should not be included in the message, if possible.	68
JMS_TIBCO_IMPORTED	Set by the server when the message has been imported from TIBCO Rendezvous.	86 111
JMS_TIBCO_MSG_EXT	Extends the functionality of map messages to include submessages or arrays.	69
JMS_TIBCO_MSG_TRACE	Specifies the message should be traced from producer to consumer.	231
JMS_TIBCO_PRESERVE_UNDELIVERED	Specifies the message is to be placed on the undelivered message queue if the message must be removed.	67

Table 1 Summary of message properties (Sheet 2 of 2)

Property	Description	More Info
JMS_TIBCO_SENDER	Contains the user name of the message sender.	68

The JMS standard specifies two delivery modes for messages, `PERSISTENT` and `NON_PERSISTENT`. TIBCO Enterprise Message Service also includes `RELIABLE_DELIVERY`. This delivery mode eliminates some of the overhead associated with the other delivery modes.

For consumer sessions, you can also specify that consumers do not need to acknowledge receipt of messages, if desired.

See Chapter 4, *Working With Messages*, on page 55 for more information about working with messages. Also, more information about properties specific to TIBCO Enterprise Message Service can be found in the *TIBCO Enterprise Message Service Java API Reference* included in the online documentation.

String Encoding

TIBCO Enterprise Message Service Java and C clients can use the functions and libraries provided for handling strings with specific character encodings. This is important for applications handling international data or any non-ASCII characters. See *Character Encoding in Messages* on page 60 for more information about character encoding in TIBCO Enterprise Message Service clients.

Message Tracing and Monitoring

Administrators can configure the server to trace messages as they are processed. Message information can be written to the log file or to the console. Message monitoring topics can also be used to receive messages that provide details about each message’s processing. See Chapter 10, *Monitoring Server Activity*, on page 225 for more information about monitoring messages or tracing message activity.

Sample Code

TIBCO Enterprise Message Service includes several example programs. These examples illustrate various features of TIBCO Enterprise Message Service. You may wish to view these example programs when reading about the corresponding features in this manual. The examples are included in the `samples` subdirectory of the TIBCO Enterprise Message Service installation directory.

For more information about running the examples, see Appendix A, Using the Samples, on page 327.

Message Priority

The JMS specification includes a message header field in which senders can set the priority of a message, as a value in the range [0,9]. EMS *does* support message priority (though it is optional, and other vendors might not implement it).

When the EMS server has several messages ready to deliver to a consumer client, and must select among them, then it delivers messages with higher priority before those with lower priority.

However, priority ordering applies only when the server has a *backlog* of deliverable messages for a consumer. In contrast, when the server rarely has only one message at a time to deliver to a consumer, then the priority ordering feature will not be apparent.

See Also JMS Specification, chapter 3.4.10

Administration

TIBCO Enterprise Message Service provides mechanisms for administering server operations and creating objects that are managed by the server, such as ConnectionFactories and Destinations; see Chapter 2, Working With the Client API, on page 19.

Administration functions can be issued either using the command-line administration tool or by creating an application that uses the administration API (either Java or .NET). The command-line administration tool is described in Chapter 8, Using the Administration Tool, on page 161. The administration APIs are described in the online documentation.

The administration interfaces allow you to create and manage administered objects such as ConnectionFactories, Topics, and Queues. EMS clients can retrieve references to these administered objects by using Java Naming and Directory Interface (JNDI). Creating static administered objects allows clients to use these objects without having to implement the objects within the client.

Administering the Server

The TIBCO Enterprise Message Service has several administration features that allow you to monitor and manage the server. The following table provides a summary of administration features and details where in the documentation you can find more information.

Table 2 Summary of administration features (Sheet 1 of 2)

Feature	More Information
Configuration files allow you to specify server characteristics.	Chapter 7, Using the Configuration Files, on page 117
Administration tool provides a command line interface for managing the server.	Chapter 8, Using the Administration Tool, on page 161
Authentication and permissions can restrict access to the server and to destinations. You can also specify who can perform administrative activities with administrator permissions.	Chapter 9, Authentication and Permissions, on page 199
Log files can be configured to provide information about various server activity.	Chapter 10, Monitoring Server Activity, on page 225

Table 2 Summary of administration features (Sheet 2 of 2)

Feature	More Information
The server can publish messages when various system events occur. This allows you to create robust monitoring applications that subscribe to these system monitor topics.	Chapter 10, Monitoring Server Activity, on page 225
The server can provide various statistics at the desired level of detail.	Chapter 10, Monitoring Server Activity, on page 225

User and Group Management

TIBCO Enterprise Message Service provides facilities for creating and managing users and groups locally for the server. The TIBCO Enterprise Message Service server can also use an external system, such as an LDAP server for authenticating users and storing group information. See Chapter 9, Authentication and Permissions, on page 199 for more information about configuring TIBCO Enterprise Message Service to work with external systems for user and group management.

Using TIBCO Hawk

You can use TIBCO Hawk for monitoring and managing the TIBCO Enterprise Message Service server. See Appendix B, Using TIBCO Hawk, on page 335 for more information.

Fault Tolerance

You can configure TIBCO Enterprise Message Service servers as primary and backup servers to provide fault tolerance for your environment. The primary and backup servers act as a pair, with the primary server accepting client connections and performing the work of handling messages, and the secondary server acting as a backup in case of failure. When the active server fails, the backup server assumes operation and becomes the primary active server.

See Chapter 13, Fault Tolerance, on page 289 for more information about the fault-tolerance features of TIBCO Enterprise Message Service.

Routing

TIBCO Enterprise Message Service provides the ability for servers to route messages between each other. Topic messages can be routed across multiple hops, provided there are no cycles (that is, the message can not be routed to any server it has already visited). Queue messages can travel at most one hop to any other server from the server that owns the queue.

TIBCO Enterprise Message Service stores and forwards messages in most situations to provide operation when a route is not connected.

See Chapter 14, *Working With Routes*, on page 303 for more information about the routing features of TIBCO Enterprise Message Service.

SSL

Secure Sockets Layer (SSL) is a protocol for transmitting encrypted data over the Internet or an internal network. SSL works by using public and private keys to encrypt data that is transferred over the SSL connection. Most web browsers support SSL, and many Web sites and Java applications use the protocol to obtain confidential user information, such as credit card numbers.

TIBCO Enterprise Message Service supports SSL. SSL is supported between the following components:

- between an EMS client and the TIBCO Enterprise Message Service server
- between the administration tool and the TIBCO Enterprise Message Service server
- between routed servers
- between fault-tolerant servers

The TIBCO Enterprise Message Service server and the EMS C client library uses OpenSSL for SSL support. You can find out more about OpenSSL at www.openssl.org.

Microsoft's .NET does not support SSL.

See Chapter 12, *Using the SSL Protocol*, on page 265 for more information about SSL support in TIBCO Enterprise Message Service.

Integrating With Third-Party Products

TIBCO Enterprise Message Service allows you to work with third-party naming/directory service products or with third-party application servers; see *TIBCO Enterprise Message Service Application Integration Guide*.

Transaction Support

TIBCO Enterprise Message Service can integrate with Java Transaction API (JTA) compliant transaction managers. TIBCO Enterprise Message Service implements all interfaces necessary to be JTA compliant.

Chapter 2

Working With the Client API

This chapter gives an introduction to working with the interfaces of the `javax.jms` package. For more information about the interfaces of this package, see *TIBCO Enterprise Message Service Java API Reference*.

Topics

- *API Overview, page 20*
- *ConnectionFactory, page 25*
- *Connection, page 26*
- *Session, page 27*
- *MessageProducer, page 28*
- *MessageConsumer, page 29*
- *MessageListener, page 30*

API Overview

Java applications use the `javax.jms` package to send or receive messages. This is a standard set of interfaces, specified by the JMS specification, for creating the connection to the EMS server, specifying the type of message to send, and creating the destination (topic or queue) to send to or receive from. You can find a description of the `javax.jms` package in *TIBCO Enterprise Message Service Java API Reference* included in the online documentation.

The JMS specification also allows vendor-specific implementations of several features. TIBCO Enterprise Message Service provides a package containing classes and constants for all TIBCO-specific aspects of TIBCO Enterprise Message Service. See the description of the `com.tibco.tibems` package in *TIBCO Enterprise Message Service Java API Reference* included in the online documentation.

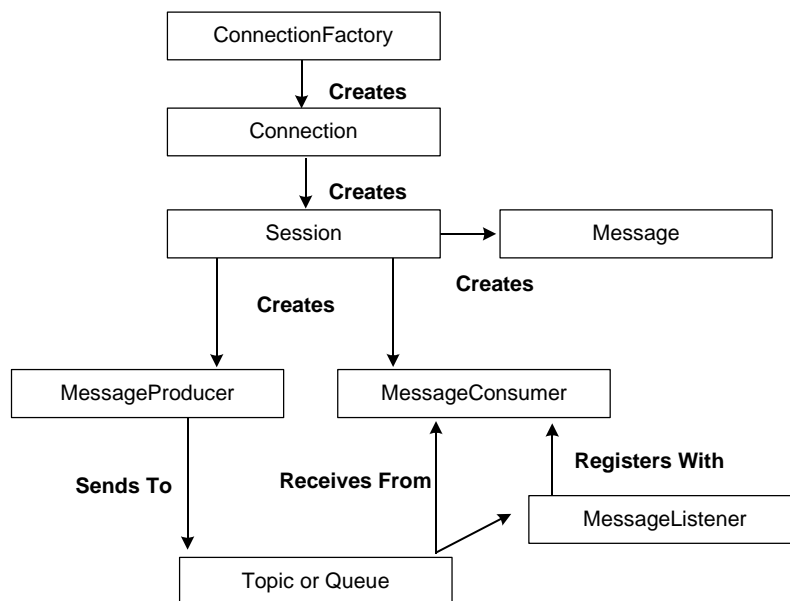
C and C# Client APIs

TIBCO Enterprise Message Service includes parallel APIs for clients written in C or in C#. For details, see *TIBCO Enterprise Message Service C & COBOL API Reference* (online documentation), and *TIBCO Enterprise Message Service .NET API Reference*.

Programming Model

Figure 4 illustrates the interfaces involved in the EMS API.

Figure 4 JMS API programming model



Applications using the point to point (queues) or publish and subscribe (topics) models use the same interfaces to create objects. The JMS specification refers to these interfaces as *common facilities* because these interfaces create objects that can be used for either topics or queues.

Previous versions of the JMS specification defined specific interfaces for topics and for queues. While these interfaces continue to be supported, they may be deprecated in future releases of the JMS specification. You should use the common facilities in your new EMS applications and upgrade old applications to use the common facilities at your earliest convenience.

The JMS API interfaces prior to the JMS 1.1 specification have the same structure as the common facilities, but the interfaces are specific to topics or queues. Figure 5 illustrates the previous interface model used by the JMS API.

Figure 5 Specific interfaces for topics and queues

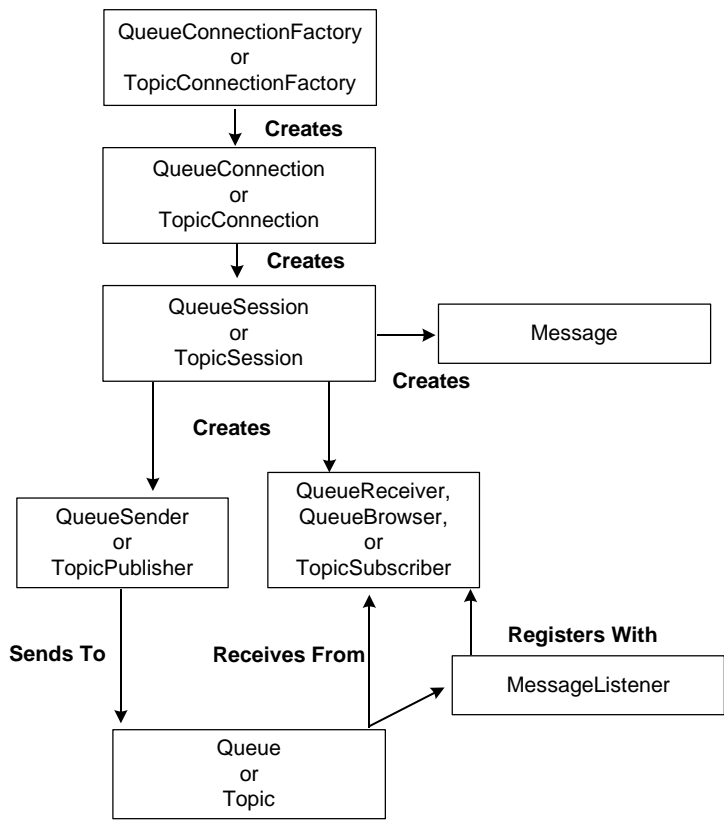


Table 3 summarizes the interfaces used in the JMS API.

Table 3 API object summary (Sheet 1 of 2)

Common Facilities Interfaces	Specific Interfaces	Description
ConnectionFactory	QueueConnectionFactory TopicConnectionFactory	Object used to create connections to EMS server.
Connection	QueueConnection TopicConnection	A connection encapsulates a physical connection with a provider (server). Connections are used to create sessions.

Table 3 API object summary (Sheet 2 of 2)

Common Facilities Interfaces	Specific Interfaces	Description
Session	QueueSession TopicSession	<p>A session is a single-threaded object that creates instances of message producers, message consumers, messages and transacted message groups.</p> <p>Sessions can also be transacted. In a transacted session, a group of messages are sent and received in a single transaction.</p>
MessageProducer	QueueSender TopicPublisher	A message producer is an object created by a session that is used for sending messages to a destination.
MessageConsumer	QueueBrowser QueueReceiver TopicSubscriber	A message consumer is an object created by a session that receives messages sent to a destination.
MessageListener		A message listener is an object that acts as an asynchronous event handler for messages. Message listeners must be registered with a specific MessageConsumer.
MessageSelector		<p>Message selectors are optional filters that can be used by the application. They transfer the filtering work to the message provider, rather than the message consumer.</p> <p>A message selector is a String that contains an expression. The syntax of the expression is based on a subset of the SQL92 conditional expression syntax.</p>
Message		Several types of message bodies are available for queues and topics.
Queue Topic	Queue Topic	<p>The destination that messages can be sent to or received from.</p> <p>Normally these are created and managed by the server, but clients can create destinations dynamically by using methods on the Session object.</p>

The following sections describe many of the API interfaces. Queues and Topics are described in Chapter 3, *Working With Destinations*, on page 31. Messages are described in Chapter 4, *Working With Messages*, on page 55.

ConnectionFactory

The ConnectionFactory object encapsulates a set of connection configuration parameters. These objects are created using the administration interface and they are stored and managed by the TIBCO Enterprise Message Service server. See Using the Administration Tool on page 161 for more information about creating and managing connection factories.

When a client starts, it typically performs a Java Naming and Directory Interface (JNDI) lookup for the ConnectionFactories that it needs. For example, the following code retrieves the InitialContext using the JNDI properties specified by env, then looks up a ConnectionFactory named myConnectionFactory.

```
Context ctx = new InitialContext(env);
ConnectionFactory myConnectionFactory =
    (ConnectionFactory) ctx.lookup("myConnectionFactory");
```

See Using JNDI with TIBCO Enterprise Message Service on page 255 for more information about using JNDI with TIBCO Enterprise Message Service.

Connection

A `Connection` object encapsulates a virtual connection with the server. `ConnectionFactory` objects create `Connection` objects. You use a `Connection` to create one or more `Session` objects. For example, using the `myConnectionFactory` object created in `ConnectionFactory` on page 25, the following creates a `Connection`:

```
Connection myConnection =  
    myConnectionFactory.createConnection()
```

A connection is a fairly heavyweight object, and therefore most clients will use one connection for all messaging. You may create multiple connections, if needed by your application.

Before consuming messages, the application must call the `start()` method on the connection. See `MessageConsumer` on page 29 for more details about `MessageConsumers`. If you wish to temporarily suspend message delivery, call the `stop()` method on the connection.

When a client application completes, all open connections must be closed. Unused open connections are eventually closed, but they do consume resources that could be used for other applications. Closing a connection also closes any `Sessions` created by the `Connection`. To close a connection, use the `close()` method. For example:

```
myConnection.close();
```

Session

A Session is a single-threaded context for producing or consuming messages. You create MessageProducers or MessageConsumers using Session objects. For example, using the `myConnection` object created in Connection on page 26, the following creates a Session:

```
Session mySession =  
    myConnection.createSession(false, Session.AUTO_ACKNOWLEDGE);
```

The first parameter to the `CreateSession()` method determines whether the Session is transactional or not. The second parameter specifies the acknowledge mode of messages received by the session. See Message Acknowledgement on page 66 and No-Acknowledgement Message Receipt on page 71 for more information about message acknowledgment modes supported in TIBCO Enterprise Message Service.

MessageProducer

A MessageProducer object is created by a Session object and is used for sending messages to destinations. For example, using the `mySession` object created in Session on page 27, the following creates a MessageProducer that sends messages to a queue named `myQueue`:

```
MessageProducer myQueueSender = mySession.createProducer(myQueue);
```

Once you have created a MessageProducer, you can use it to send messages. See Chapter 4, Working With Messages, on page 55 for more information about creating messages. For example, the following sends a message using the `queueSender` created above:

```
myQueueSender.send(message);
```

You can create MessageProducers that do not identify a destination. When the sender or publisher does not specify a destination, you must specify the destination when you send or publish a message as the first parameter of the `send()` or `publish()` method.

MessageConsumer

A MessageConsumer object is created by a Session object and is used for receiving messages sent to destinations. For example, using the `mySession` object created in Session on page 27, the following creates a MessageConsumer that retrieves messages from a queue named `myQueue`:

```
MessageConsumer myQueueReceiver =
    mySession.createConsumer(myQueue);
```

For queues, messages remain on the queue until they are consumed by a MessageConsumer, the message expiration time has been reached, or the maximum size of the queue is reached.

The following sections describe how message consumers can obtain messages.

Durable Subscribers for Topics

Only MessageConsumers whose client applications are running receive messages published on a topic. Optionally, Sessions can create durable subscribers to ensure that messages are received, even if the application is not currently running.

Use the `Session.createDurableSubscriber()` method to create a durable subscription. Messages are stored by the server as long as durable subscribers exist for the topic, or until the message expiration time for the message has been reached, or until the storage limit has been reached for the topic.

When an application restarts and recreates a durable subscriber with the same ID, all messages stored on the server for that topic are published to the durable subscriber.

For more information on using the `createDurableSubscriber()` method, see the *TIBCO Enterprise Message Service Java API Reference*.

Synchronous or Asynchronous Message Consumption

The API allows for synchronous or asynchronous message consumption. For synchronous consumption, the MessageConsumer explicitly calls the `receive()` method on the topic or queue. For asynchronous consumption, the client registers a MessageListener for the topic or queue. When a message arrives at the destination, the TIBCO Enterprise Message Service server delivers the message by calling the listener's `onMessage()` method.

MessageListener

A `MessageListener` object acts as an asynchronous event handler for messages. This object implements the `MessageListener` interface and has one method, `onMessage()`.

The `onMessage()` method is called by the TIBCO Enterprise Message Service server when a message arrives on a destination. You implement the `onMessage()` method in your `MessageListener` class to perform the desired actions when a message arrives. Your implementation should handle all exceptions, and it should not throw any exceptions.

Once you create a `MessageListener` object, you must register it with a specific `MessageConsumer`. For example, using the `myQueueReceiver` object created in `MessageConsumer` on page 29, the following creates a `queueListener` (an implementation of the `MessageListener` interface) and registers it with the `QueueReceiver` object:

```
MessageListener queueListener = new MessageListener();  
myQueueReceiver.setMessageListener(queueListener);
```

You should register the `MessageListener` with the `MessageConsumer` before calling the `Connection`'s `start()` method to begin receiving messages.

A `MessageListener` is not specific to the type of the destination. The same listener can obtain messages from a queue or a topic depending upon whether the `MessageConsumer` that registered the listener is a `TopicSubscriber` or a `QueueReceiver`.

The J2EE 1.3 platform introduced message-driven beans (MDBs) that are a special kind of `MessageListener`. See the J2EE documentation for more information about MDBs.

Chapter 3 **Working With Destinations**

This chapter describes destinations (topics and queues).

Topics

- *Destination Overview, page 32*
- *Destination Properties, page 34*
- *Wildcards, page 44*
- *Inheritance, page 46*
- *Destination Bridges, page 48*
- *Flow Control, page 51*

Destination Overview

Destinations for messages can be either Topics or Queues. A destination can be created statically in the server configuration files, or dynamically by a client application.

Static Queues and Topics

Configuration information for static queues and topics is stored in configuration files for the TIBCO Enterprise Message Service server. Changes to the configuration information can be made in a variety of ways. To manage static destinations, you can edit the configuration files using a text editor, you can use the administration tool, or you can use the administration APIs.

Static queues and topics are administered by the server. Clients retrieve the destination using JNDI. For example, the following code retrieves the InitialContext using the JNDI properties specified by `env`, then looks up a Topic named `MyTopic`.

```
Context ctx = new InitialContext(env);
Topic myTopic = (Topic) ctx.lookup("MyTopic");
```

See [Using JNDI with TIBCO Enterprise Message Service](#) on page 255 for more information about using JNDI with TIBCO Enterprise Message Service.

Dynamic Queues and Topics

Dynamic queues and topics are created on-the-fly by applications using `QueueSession.createQueue()` and `TopicSession.createTopic()`. Dynamic queues and topics do not appear in the configuration files, and exist as long as there are messages or consumers on the destination. A client cannot use JNDI to lookup dynamic queues and topics.

When you use the `show queues` or `show topics` command in the administration tool, you see both static and dynamic topics and queues. The dynamic topics and queues have an asterisk (*) in front of their name in the list of topics or queues.

Dynamic queues and topics inherit properties from their respective parents. When shown by the administration tool, properties of a queue or topic may have an asterisk (*) character in front of its name. This means that this property was inherited from the parent queue or topic and cannot be changed. For more information, refer to [Inheritance of Properties](#) on page 46 and [Wildcards *](#) and [>](#) on page 44.

Temporary Queues and Topics

TIBCO Enterprise Message Service supports temporary queues and topics as defined in JMS specification 1.1 and its API.

Servers connected by routes exchange messages sent to temporary topics. As a result, temporary topics are ideal destinations for reply messages in request/reply interactions.

For more information on temporary queues and topics, refer to the JMS documentation described in Third Party Documentation on page xviii.

Destination Bridges

You can create server-based bridges between destinations. This allows all messages delivered to one destination to also be delivered to the bridged destination. You can bridge between different destination types, between the same destination type, or to more than one destination. For example, you can create a bridge between a topic and a queue or from a topic to another topic.

See Destination Bridges on page 48 for more information about destination bridging.

Destination Properties

This section contains a description of properties for topics and queues. Table 4 lists the properties that can be assigned to topics and queues. The following sections describe each property.

Table 4 Destination properties (Sheet 1 of 2)

Property	Described on Page	Topic	Queue
failsafe	35	Yes	Yes
secure	36	Yes	Yes
maxbytes	36	Yes	Yes
maxmsgs	37	No	Yes
overflowPolicy	37	No	Yes
global	37	Yes	Yes
sender_name	37	Yes	Yes
sender_name_enforced	38	Yes	Yes
flowControl	38	Yes	Yes
trace	39	Yes	Yes
import	39	Yes	Yes
export	39	Yes	No
maxRedelivery	40	No	Yes
exclusive	40	No	Yes
prefetch	40	No	Yes
expiration	42	Yes	Yes

Deprecated Properties

The following properties are available for backward-compatibility with previous versions. The functionality of these properties is replaced with the `import` and `export` properties.

Table 4 Destination properties (Sheet 2 of 2)

Property	Described on Page	Topic	Queue
tibrv_import	–	Yes	Yes
tibrvcm_import	–	Yes	Yes
tibrv_export	–	Yes	No
tibrvcm_export	–	Yes	No

failsafe

TIBCO Enterprise Message Service provides two modes for persisting topic/queue messages in external storage. These two modes are:

- normal
- failsafe

Normal mode writes all messages into the file on disk in asynchronous mode. In this mode, the data may remain in system buffers for a short time before it is written to disk.

Asynchronous mode storage includes a small probability that, in case of software or hardware failure, some data may be lost without the possibility of recovery. In many applications, when a rare loss of a few messages is acceptable, this mode provides the best combination of performance and reliability.

For situations in which any loss of data is not acceptable, the administrator should set the `failsafe` property for the topic or the queue. In failsafe mode, all data for that queue or topic are written into external storage in synchronous mode. In synchronous mode, a write operation is not complete until the data is physically recorded on the external device.

The `failsafe` property ensures that no messages are ever lost in case of server failure. Although failsafe mode guarantees no messages are lost, it also significantly affects the performance.

secure

When the `secure` property is enabled for a destination, it instructs the server to check user permissions whenever a user attempts to perform an operation on that destination. When the `secure` property is disabled for a destination, the server does not check permissions for that destination—any authenticated user can perform any operations on that topic or queue.



The `secure` property is independent of SSL—it controls basic authentication and permission verification within the server. To configure secure communication between clients and server, see *Using the SSL Protocol* on page 265.

The server `authorization` property acts as a master switch for checking permissions. That is, the server checks user permissions on secure destinations only when the `authorization` property is enabled. To enforce permissions, you must *both* enable the `authorization` configuration parameter, and set the `secure` property on each affected destination.

See Chapter 9, *Authentication and Permissions*, on page 199 for more information on permissions and the `secure` property.

maxbytes

Topics and queues can specify the `maxbytes` property in the form: `maxbytes=b` where *b* is the number of bytes.

For queues, `maxbytes` defines the maximum size (in bytes) that the queue can store, summed over all messages in the queue. When adding a message would exceed this limit, the server does not accept the message into storage, and the message producer's `send` call returns an error (but see also `overflowPolicy`). For example, if a receiver is off-line for a long time, then the queue size could reach this limit, which would prevent further memory allocation for additional messages.

If `maxbytes` is zero, or is not set, the server does not limit the memory allocation for the queue.

You can set both `maxmsgs` and `maxbytes` properties on the same queue. Exceeding either limit causes the server to reject new messages until consumers reduce the the queue size to below these limits.

For topics, `maxbytes` limits the maximum size (in bytes) that the topic can store for delivery to each durable subscriber on that topic. That is, the limit applies separately to each durable subscriber on the topic. For example, if a durable subscriber is off-line for a long time, pending messages accumulate until they exceed `maxbytes`; when the subscriber consumes messages (freeing storage) the topic can accept additional messages for the subscriber.

When a destination inherits different values of this property from several parent destinations, it inherits the smallest value.

maxmsgs

Queues can specify the `maxmsgs` property in the form: `maxmsgs=m` where *m* is the number of messages.

`maxmsgs` defines the maximum number of messages that can be waiting in a queue. When adding a message would exceed this limit, the server does not accept the message into storage, and the message producer's `send` call returns an error (but see also `overflowPolicy`).

If `maxmsgs` is zero, or is not set, the server does not limit the number of messages in the queue.

You can set both `maxmsgs` and `maxbytes` properties on the same queue. Exceeding either limit causes the server to reject new messages until consumers reduce the the queue size to below these limits.

overflowPolicy

Queues can specify the `overflowPolicy` property to change the effect of exceeding queue limits (either `maxbytes` or `maxmsgs`).

When `overflowPolicy=discardOld`, exceeding queue limits causes the server to discard the oldest messages in the queue to free space for new messages from producers.

When `overflowPolicy` is not set, the server refuses new messages for the queue, and the producer `send` call indicates an error.

global

Messages destined for a topic or queue with the `global` property set are routed to the other servers that are participating in routing with this server. For further information on routing between servers, see Chapter 14, *Working With Routes*, on page 303.

sender_name

The `sender_name` property specifies that the server may include the sender's username for messages sent to this destination. When this property is enabled, the server takes the user name supplied by the message producer when the connection is established and places that user name into the `JMS_TIBCO_SENDER` property in the message.

The message producer can override this behavior by specifying a property on a message. If a message producer sets the `JMS_TIBCO_DISABLE_SENDER` property to true for a message, the server overrides the `sender_name` property and does not add the sender name to the message.

If authentication for the server is turned off, the server places whatever user name the message producer supplied when the message producer created a connection to the server. If authentication for the server is enabled, the server authenticates the user name supplied by the connection and the user name placed in the message property will be an authenticated user. If SSL is used, the SSL connection protocol guarantees the client is authenticated using the client's digital certificate.

sender_name_enforced

The `sender_name_enforced` property specifies that messages sent to this destination *must* include the sender's user name. The server retrieves the user name of the message producer using the same procedure described in the `sender_name` property above. However, unlike, the `sender_name` property, there is no way for message producers to override this property.

If the `sender_name` property is also set on the destination, this property overrides the `sender_name` property.



In some business situations, clients may not be willing to disclose the username of their message producers. If this is the case, these clients may wish to avoid sending messages to destinations that have the `sender_name` or `sender_name_enforced` properties enabled.

In these situations, the operator of the EMS server should develop a policy for disclosing a list of destinations that have these properties enabled. This will allow clients to avoid sending messages to destinations that would cause their message producer usernames to be exposed.

flowControl

The `flowControl` property specifies the target maximum size the server can use to store pending messages for the destination. This is useful when message producers send messages much more quickly than message consumers can consume them. Using this property can prevent the pending messages from consuming too many machine resources.

The value specified for this property is in bytes, unless you specify the units for the amount of storage using KB, MB, or GB. For example, `flowControl=60MB` specifies the target maximum storage for pending messages of the destination is 60 Megabytes. You can specify the `flowControl` property without a value to set it to the default of 256KB.

Flow control must be enabled for the server before the value in this property is enforced by the server. See Flow Control on page 51 for more information about flow control.

trace

Specifies that tracing should be enabled for this destination. This property can be specified as either `trace` or `trace=body`. Specifying `trace` (without `=body`), generates trace messages that include only the message sequence and message ID. Specifying `trace=body` generates trace messages that include the message body. See Message Tracing on page 231 for more information about message tracing.

import

The `import` property allows messages published by an external system to be received by a TIBCO Enterprise Message Service destination (a topic or a queue), as long as the transport to the external system is configured. Currently you can configure transports for TIBCO Rendezvous reliable and certified messaging protocols. You can specify the name of one or more transports of the same type in the `import` property.

You must purchase, install, and configure the external system (for example, Rendezvous) before configuring topics with the `import` property. Also, you must configure the communication parameters to the external system by creating a named transport in the `transports.conf` file.

For complete details about external message services, see these chapters:

- Chapter 5, Working With TIBCO Rendezvous, on page 73
- Chapter 6, Working With TIBCO SmartSockets, on page 95

export

The `export` property allows messages published by a client to a topic to be exported to the external systems with configured transports. Currently you can configure transports for Rendezvous reliable and certified messaging protocols. You can specify the name of one or more transports of the same type in the `export` property.

You must purchase, install, and configure the external system (for example, Rendezvous) before configuring topics with the `export` property. Also, you must configure the communication parameters to the external system by creating a named transport in the `transports.conf` file.

For complete details about external message services, see these chapters:

- Chapter 5, Working With TIBCO Rendezvous, on page 73

- Chapter 6, Working With TIBCO SmartSockets, on page 95

maxRedelivery

The `maxRedelivery` property specifies the number of attempts the server should make to redeliver a message sent to a queue. The value of this parameter can be set to an integer between 2 and 255. Once the server has attempted to deliver the message the specified number of times, the message is either destroyed or it is placed on the undelivered queue, if the `JMS_TIBCO_PRESERVE_UNDELIVERED` property on the message is set to `true`.

For messages that have been redelivered, the `JMSRedelivered` header property is set to `true` and the `JMSXDeliveryCount` property is set to the number of times the message has been delivered to the queue. If the server restarts, the current number of delivery attempts in the `JMSXDeliveryCount` property is not retained.

For more information about undelivered messages, see Undelivered Message Queue on page 67.

exclusive

The `exclusive` property is available for queues only (not for topics). When `exclusive` is set to `true`, the server sends all messages (on that queue) to one consumer. Any other consumers do not receive any messages from the queue. Instead, these additional consumers act in a *standby* role; if the primary consumer fails, the server selects one of the standby consumers as the new primary, and begins delivering messages to it.

Non-Exclusive Queues & Round-Robin Delivery

With non-exclusive queues (`exclusive` set to `false`) the server distributes messages in a round-robin—one to each receiver that is ready. If any receivers are still ready to accept additional messages, the server distributes another round of messages—one to each receiver that is still ready. When none of the receivers are ready to receive more messages, the server waits until a queue receiver reports that it can accept a message.

This arrangement prevents a large buildup of messages at one receiver and balances the load of incoming messages across a set of queue receivers.

prefetch

The `prefetch` property applies only to queues.

Background

Delivering messages from the server to a client program involves two independent phases—fetch and accept:

- The *fetch* phase is a two-step interaction between a `MessageConsumer` object (in a client program) and the server.
 - The `MessageConsumer` initiates the fetch phase by signalling to the server that it is ready for more messages.
 - The server responds by transferring one or more messages to the client, which stores them in the `MessageConsumer` object.
- In the *accept* phase, client code takes a message from the `MessageConsumer` object.

The `receive` call embraces both of these phases. It initiates fetch when needed, and it accepts a message from the `MessageConsumer` object.

To reduce waiting time for client programs, the `MessageConsumer` can *prefetch* messages—that is, fetch a batch of messages from the server, and hold them for client code to accept, one by one.

Values The `MessageConsumer` and the server cooperate to regulate fetching according to the queue's prefetch property. Table 5 presents the range of values.

Table 5 Prefetch

Value	Description
2 or more	The <code>MessageConsumer</code> automatically fetches messages from the server. The <code>MessageConsumer</code> never stores more than this maximum number of messages.
1	The <code>MessageConsumer</code> automatically fetches messages from the server—initiating fetch only when it does not currently hold a message.
none	Disables automatic fetch. That is, the <code>MessageConsumer</code> initiates fetch only when the client calls <code>receive</code> —either an explicit synchronous call, or an implicit call (in an asynchronous receiver).
0	<p>The queue inherits the prefetch value. If it has no parent, or no queue in the parent chain sets a value for prefetch, then the default value is 5.</p> <p>When a queue does not set any value for prefetch, then the default value is 0 (that is, inherit the prefetch value).</p>

Automatic Fetch Enabled	<p>To enable automatic fetch, set <code>prefetch</code> to a positive integer. Automatic fetch ensures that a message is always waiting when client code is ready to accept one. It can improve performance by decreasing or eliminating client idle time while the server transfers a message.</p> <p>However, when a <code>MessageConsumer</code> prefetches a group of messages, the server does not deliver them to other consumers (unless the first consumer's connection to the server is broken).</p>
Automatic Fetch Disabled	<p>To disable automatic fetch, set <code>prefetch=none</code>.</p> <p>Even when <code>prefetch=none</code>, a <code>MessageConsumer</code> object can still hold a message. For example, a <code>receive</code> call initiates fetch, but its timeout elapses before the server finishes transferring the message. This situation leaves a fetched message waiting in the <code>MessageConsumer</code>. A second <code>receive</code> call does not fetch another message; instead, it accepts the message that is already waiting. A third <code>receive</code> call initiates another fetch.</p> <p>Notice that a waiting message still belongs to the <code>MessageConsumer</code>, and the server does not deliver it to another consumer (unless the first consumer's connection to the server is broken). To prevent messages from waiting in this state for long periods of time, code programs either to call <code>receive</code> with no timeout, or to call it (with timeout) repeatedly and shorten the interval between calls.</p>
Inheritance	<p>When a queue inherits the <code>prefetch</code> property from parent queues with matching names, these behaviors are possible:</p> <ul style="list-style-type: none"> • When all parent queues set the value <code>none</code>, then the child queue inherits the value <code>none</code>. • When any parent queue sets a non-zero numeric value, then the child queue inherits the <i>largest</i> value from among the entire parent chain. • When none of the parent queues sets any non-zero numeric value, then the child queue uses the default value (which is 5).

expiration

The server's `expiration` property overrides expiration values set by message producers (in client programs). You can set this property for any queue and any topic.

If this property is set for a destination, then when the server delivers a message to that destination, the server replaces the producer's expiration value with this value.

Specify this value as an integer with units. Legal units are msec, sec, min, hour and day (for example, `expiration=10min`). When units are absent, the default unit is seconds.

Zero is a special value, which indicates that messages to the destination never expire.

Whenever a client application uses non-zero values for message expiration, 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 message expiration time.

Wildcards

Static queues and topics are assigned certain properties in the configuration file. These static queues and topics become the parents of dynamic queues and topics, which inherit properties from the parents. You must understand wildcards to understand the inheritance rules.

Wildcards * and >

To understand the rules for inheritance of properties, it is important to understand the use of the two wildcards, * and >.

- The wildcard * means that any token can be in the place of *

For example: `foo.*` matches all two-part destination names beginning with `foo.` including `foo.bar` and `foo.booo`, but not `foo.bar.booo`.

However, `foo.*.bar` matches all three-part destination names with a token in the wildcard position. In this case, `foo.booo.bar` is matched, but `foo.bar` is not.

- The wildcard > matches one or more trailing elements.

For example, `foo.>` matches `foo.bar` and `foo.bar.booo`

Wildcards in Topics

TIBCO Enterprise Message Service enables you to use wildcards in topic names in some situations:

- You can subscribe to wildcard topics.

If you subscribe to a topic containing a wildcard, you will receive any message published to a matching topic. For example, if you subscribe to `foo.*` you will receive messages published to a topic named `foo.bar`.

You can subscribe to a wildcard topic (for example `foo.*`), whether or not there is a matching topic in the configuration file (for example, `foo.*`, `foo.>`, or `foo.bar`). However, if there is no matching topic name in the configuration file, no messages will be published on that topic, so it is not useful to subscribe to the wildcard topic in that case.

- You cannot publish to wildcard topics.
- If `foo.bar` is not in the configuration file, then you can publish to `foo.bar` if `foo.*` or `foo.>` exists in the configuration file.

Wildcards in Queues

TIBCO Enterprise Message Service enables you to use wildcards in queue names in some situations. You can not send or receive to wildcard queue names. However, you can use wildcard queue names in the configuration files. The wildcard queue names in the configuration files must have non-wildcard children to send and receive messages.

For example, if the queue configuration file includes a line:

```
foo.*
```

then users can create queues `foo.bar`, `foo.bob`, and so forth, but not `foo.bar.bob`.

Inheritance

This section describes the inheritance of properties and permissions. For more information on wildcards, refer to Wildcards on page 44. For more information on properties, refer to Destination Properties on page 34. For more information on permissions, refer to Chapter 9, Authentication and Permissions, on page 199.

Inheritance of Properties

All properties are inheritable for both topics and queues. This means that a property set for a destination is inherited by all destinations with matching names. For example, if topic `foo.*` is `failsafe`, then topics `foo.bar` and `foo.bob` inherit `failsafe` from `foo.*`.

Currently there is no way to make topic `foo.*` `failsafe` without making `foo.bar` `failsafe`. In other words, the system does not offer the ability to remove inherited properties.

Property inheritance is powerful, but complex to understand and administer. You must plan before assigning properties to topics and queues. Using wildcards to assign properties is a powerful tool, but must be used carefully.

For example, if you enter the following line in the topics configuration file:

```
> failsafe
```

you make every topic `failsafe`, regardless of additional entries. This might require a great deal of memory for storage and greatly decrease the system performance.



There is one property that is an exception to the rules of inheritance. The `maxbytes` property has the following rules of inheritance:

- If there is not a direct property value for the child, the most restrictive (smallest) of the parent or ancestor property values is used.
- The child value, which is directly assigned to the child, overrides any values assigned to ancestors.

Inheritance of Permissions

Inheritance of permissions is similar to inheritance of properties. If the parent has a permission, then the child inherits that permission. For example, if Bob belongs to GroupA, and GroupA has `publish` permission on a topic, then Bob has `publish` permission on that topic.

Permissions for a single user are the union of the permissions set for that user, and of all permissions set for every group in which the user is a member. These permission sets are additive. Permissions have positive boolean inheritance. Once a permission right has been granted through inheritance, it can not be removed.

All rules for wildcards apply to inheritance of permissions. For example, if a user has permission to publish on topic `foo.*`, the user also has permission to publish on `foo.bar` and `foo.new`. For more information on wildcards, refer to Wildcards on page 44. For more information on permissions, refer to Chapter 9, Authentication and Permissions, on page 199.

Destination Bridges

Some applications require the same message to be sent to more than one destination, possibly of different types. For example, an application may send messages to a queue for distributed load balancing. That same application, however, may also need the messages to be published to several monitoring applications. Another example is an application that publishes messages to several topics. All messages however, must also be sent to a database for backup and for data mining. A queue is used to collect all messages and send them to the database.

An application can process messages so that they are sent multiple times to the required destinations. However, such processing requires significant coding effort in the application. TIBCO Enterprise Message Service provides a server-based solution to this problem. You can create bridges between destinations so that messages sent to one destination are also delivered to all bridged destinations.

Bridges are created between one destination and one or more other destinations of the same or of different types. That is, you can create a bridge from a topic to a queue or from a queue to a topic. You can also create a bridge between one destination and multiple destinations. For example, you can create a bridge from topic `a.b` to queue `q.b` and topic `a.c`.

When specifying a bridge, you can specify a particular destination name, or you can use wildcards. For example, if you specify a bridge on topic `foo.*` to queue `foo.queue`, messages delivered to any topic matching `foo.*` are sent to `foo.queue`.

Figure 6 and Figure 7 illustrate example bridging scenarios.

Figure 6 Bridging a topic to a queue

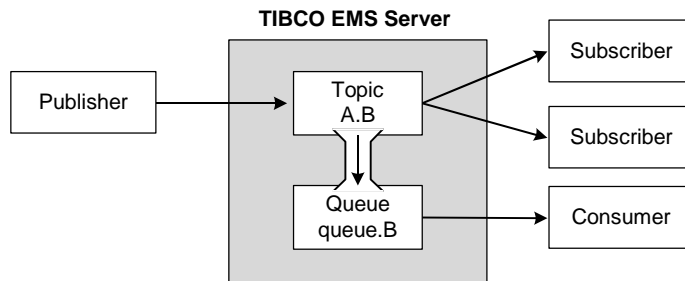
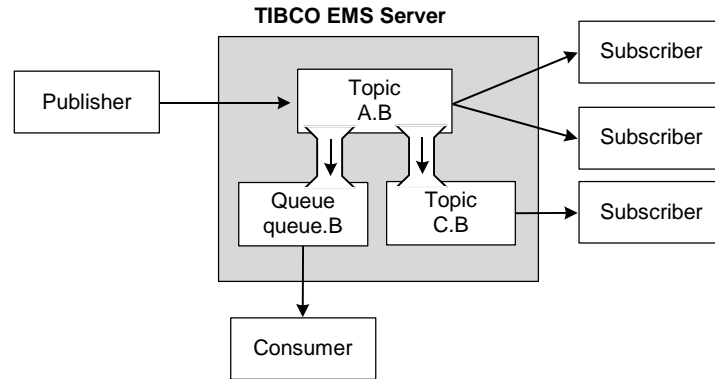


Figure 7 Bridging a topic to multiple destinations



Bridges are not transitive. That is, messages sent to a destination with a bridge are only delivered to the specified bridged destinations and are not delivered across multiple bridges. For example, topic A.B has a bridge to queue Q.B. Queue Q.B has a bridge to topic B.C. Messages delivered to A.B are also delivered to Q.B, but not to B.C.

Creating a Bridge

Bridges are configured in the `bridges.conf` configuration file. You specify a bridge using the following syntax:

```
[<destinationType> : <destinationName> ]
  <destinationType>=<destinationToBridgeTo> selector="<messageSelector>"
```

where `<destinationType>` is the type of the destination (either `topic` or `queue`), `<destinationName>` is the name of the destination from which you wish to create a bridge, `<destinationToBridgeTo>` is the name of the destination you wish to create a bridge to, and `selector="<messageSelector>"` is an optional message selector to specify the subset of messages the destination should receive.

Each `<destinationName>` can specify wildcards, and therefore any destination matching the pattern will have the specified bridge. Each `<destinationName>` can specify more than one `<destinationToBridgeTo>`.

For example, the bridge illustrated in Figure 6 and Figure 7 would be specified as the following in `bridges.conf`:

```
[topic:A.B]
  queue=queue.B
  topic=C.B
```

Specifying a message selector on a bridged destination is described in the following section.

Selecting the Messages to Bridge

By default, all messages sent to a destination with a bridge are sent to all bridged destinations. This can cause unnecessary network traffic if each bridged destination is only interested in a subset of the messages sent to the original destination. You can optionally specify a message selector for each bridge to determine which messages are sent over that bridge.

Message selectors for bridged destinations are specified as the `selector` property on the bridge. The following is an example of specifying a selector on the bridges defined in the previous section:

```
[topic:A.B]
  queue=queue.B
  topic=C.B selector="urgency in('medium', 'high')"
```

For detailed information about message selector syntax, see documentation for the `Message` class in *TIBCO Enterprise Message Service Java API Reference*.

Access Control and Bridges

Message producers must have access to a destination in order to send messages to that destination. However, a bridge automatically has permission to send to its target destination. Special configuration is not required.

Transactions

When a message producer sends a message within a transaction, all messages sent across a bridge are part of the transaction. Therefore, if the transaction succeeds, all messages are delivered to all bridged destinations. If the transaction fails, no consumers for bridged destinations receive the messages.

If a message cannot be delivered to a bridged destination because the message consumer does not have the correct permissions for the bridged destination, the transaction cannot complete, and therefore fails and is rolled back.

Flow Control

In some situations, message producers may send messages more rapidly than message consumers can receive them. The pending messages for a destination are stored by the server until they can be delivered, and the server can potentially exhaust its storage capacity if the message consumers do not receive messages quickly enough. To avoid this, TIBCO Enterprise Message Service allows you to control the flow of messages to a destination. Each destination can specify a target maximum size for storing pending messages. When the target is reached, TIBCO Enterprise Message Service blocks message producers when new messages are sent. This effectively slows down message producers until the message consumers can receive the pending messages.

Enabling Flow Control

The `flow_control` parameter in `tibemsd.conf` enables and disables flow control globally for the TIBCO Enterprise Message Service server. When `flow_control` is disabled (the default setting), the server does not enforce any flow control on destinations. When `flow_control` is enabled, the server enforces any flow control settings specified for each destination. See Chapter 7, *Using the Configuration Files*, on page 117 for more information about working with configuration parameters.

When you wish to control the flow of messages on a destination, set the `flowControl` property on that destination. The `flowControl` property specifies the target maximum size of stored pending messages for the destination. The size specified is in bytes, unless you specify the units for the size. You can specify KB, MB, or GB for the units. For example, `flowControl = 60MB` specifies the target maximum storage for pending messages for a destination is 60 Megabytes.

Enforcing Flow Control

The value specified for the `flowControl` property on a destination is a target maximum for pending message storage. When flow control is enabled, the server may use slightly more or less storage before enforcing flow control, depending upon message size, number of message producers, and other factors. Setting the `flowControl` property on a destination but specifying no value causes the server to use a default value of 256KB.

When the storage for pending messages is near the specified limit, the server blocks all new calls to send a message from message producers. The calls do not return until the storage has decreased below the specified limit. Once message consumers have received messages and the pending message storage goes below the specified limit, the server allows the send message calls to return to the caller and the message producers can continue processing.

If there are no message consumers for a destination, the server does not enforce flow control for the destination. That is, if a queue has no started receivers, the server cannot enforce flow control for that queue. Also, if a topic has inactive durable subscriptions or no current subscribers, the server cannot enforce flow control for that topic. For topics, if flow control is set on a specific topic (for example, `foo.bar`), then flow control is enforced as long as there are subscribers to that topic or any parent topic (for example, if there are subscribers to `foo.*`).

Routes and Flow Control

For global topics where messages are routed between servers, flow control can be specified for a topic on either the server where messages are produced or the server where messages are received. Flow control is not relevant for queue messages that are routed to another server.

If the `flowControl` property is set on the topic on the server receiving the messages, when the pending message size limit is reached, messages are not forwarded by way of the route until the topic subscriber receives enough messages to lower the pending message size below the specified limit.

If the `flowControl` property is set on the topic on the server sending the messages, the server may block any topic publishers when sending new messages if messages cannot be sent quickly enough by way of the route. This could be due to network latency between the routed servers or it could be because flow control on the other server is preventing new messages from being sent.

Destination Bridges and Flow Control

Flow control can be specified on destinations that are bridged to other destinations. If you wish the flow of messages sent by way of the bridge to slow down when receivers on the bridged-to destination cannot process the messages quickly enough, you must set the `flowControl` property on both destinations on either side of the bridge.

Flow Control, Threads and Deadlock



When using flow control, you must be careful to avoid potential deadlock situations.

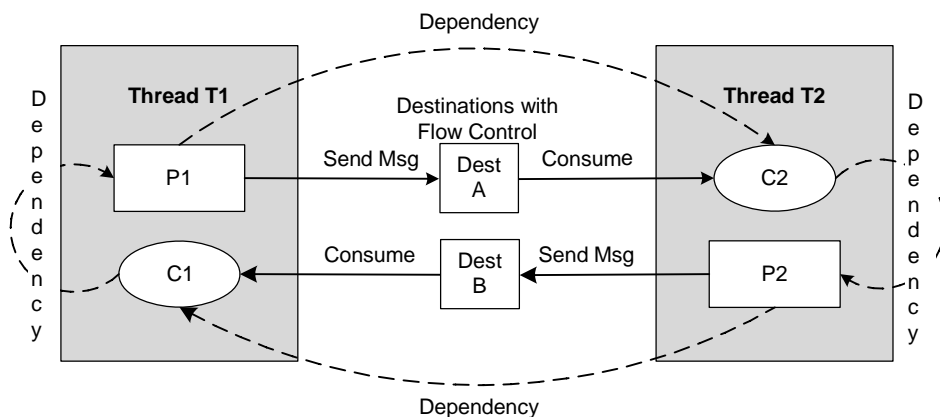
When flow control is in effect for a destination, producers to that destination can block waiting for flow control signals from the destination's consumers. If any of those consumers are within the same thread of program control, a potential for deadlock exists. Namely, the producer will not unblock until the destination contains fewer messages, and the consumer in the blocked thread cannot reduce the number of messages.

The simplest case to detect is when producer and consumer are in the same session (sessions are limited to a single thread). But more complex cases can arise. Deadlock can even occur across several threads (or even programs on different hosts), if dependencies link them. For example, consider the situation in Figure 8:

- Producer P1 in thread T1 has a consumer C2 in thread T2.
- Producer P2 in T2 has a consumer C1 in T1.
- Because of the circular dependency, deadlock can occur if either producer blocks its thread waiting for flow control signals.

The dependency analysis is analogous to mutex deadlock. You must analyze your programs and distributed systems in a similar fashion to avoid potential deadlock.

Figure 8 Flow Control Deadlock across Two Threads



Chapter 4 **Working With Messages**

This chapter describes JMS messages and how to work with them in a client program.

Topics

- *JMS Message Structure, page 56*
- *Message Persistence, page 59*
- *Character Encoding in Messages, page 60*
- *Message Compression, page 65*
- *Message Acknowledgement, page 66*
- *Undelivered Message Queue, page 67*
- *Message Extensions, page 69*
- *EMS Message Delivery Mode Extensions, page 70*

JMS Message Structure

A JMS message has the same structure, regardless of whether it is addressed to a topic or a queue. A JMS message has three sections:

- Header (some header fields are required)
- Properties (optional)
- Body (optional)

Header Fields

The header contains ten predefined fields that contain values used to route and deliver messages. Table 6 describes the message header fields.

Table 6 JMS Message Headers (Sheet 1 of 2)

Header Field	Set by	Comments
JMSDestination	send or publish method	Destination to which message is sent
JMSDeliveryMode	send or publish method	Persistent or non-persistent message
JMSExpiration	send or publish method	<p>Length of time that message will live before expiration. If set to 0, message does not expire. The time-to-live is specified in milliseconds.</p> <p>Whenever your application uses non-zero values for message expiration, 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 message expiration time.</p>
JMSPriority	send or publish method	Uses a numerical ranking, between 0 and 9, to define message priority as normal or expedited. Larger numbers represent higher priority.
JMSMessageID	send or publish method	Value uniquely identifies each message sent by a provider.

Table 6 JMS Message Headers (Sheet 2 of 2)

Header Field	Set by	Comments
JMSTimestamp	send or publish method	Timestamp of time when message was handed off to a provider to be sent. Message may actually be sent later than this timestamp.
JMSCorrelationID	message client	This ID can be used to link messages, such as linking a response message to a request message. Entering a value in this field is optional.
JMSReplyTo	message client	A destination to which a message reply should be sent. Entering a value for this field is optional.
JMSType	message client	message type identifier
JMSRedelivered	JMS provider	If this field is set, it is possible that the message was delivered to the client earlier, but not acknowledged at that time.

Properties

In the properties area, applications, vendors, and administrators on JMS systems can add optional properties. The properties area is optional, and can be left empty.

TIBCO Enterprise Message Service includes several vendor-specific properties in this area. TIBCO-specific property names begin with `JMS_TIBCO`. These properties are described in subsequent sections in this chapter.

Message Bodies

A JMS message has one of several types of message bodies, or no message body at all.

The types of messages are described in Table 7.

Table 7 JMS Message Types (Sheet 1 of 2)

Message Type	Contents of Message Body
Message	This message type has no body. This is useful for simple event notification.
TextMessage	A <code>java.lang.String</code> . For example, this can be the contents of an XML file.

Table 7 JMS Message Types (Sheet 2 of 2)

Message Type	Contents of Message Body
MapMessage	A set of name/value pairs. The names are <code>java.lang.String</code> objects, and the values are Java primitive value types or their wrappers. The entries can be accessed sequentially by enumeration or directly by name. The order of entries is undefined.
BytesMessage	A stream of uninterrupted bytes. The bytes are not typed; that is, they are not assigned to a primitive data type.
StreamMessage	A stream of primitive values in the Java programming language. Each set of values belongs to a primitive data type, and must be read sequentially.
ObjectMessage	A serializable object constructed in the Java programming language.

Maximum Message Size

EMS supports messages up to a maximum size of 512MB. However, we recommend that application programs use smaller messages, since messages approaching this maximum size will strain the performance limits of most current hardware and operating system platforms.

Message Persistence

JMS defines two message delivery modes, persistent and non-persistent. This mode is set by the message sender or publisher in the `JMSDeliveryMode` message header field. Non-Persistent messages are never written to persistent storage. Persistent messages are logged to persistent storage when they are sent.

Messages with the persistent delivery mode are always written to persistent storage, except when they are published to a topic that has no durable subscribers. When a topic has no durable subscribers, there are no subscribers that need messages resent in the event of a server failure. Therefore, messages do not need to be saved, and performance is improved because disk I/O is not required.

This behavior is consistent with the JMS specification because durable subscribers for a topic cause published messages to be saved. However, non-durable subscribers that re-connect after a server failure are considered newly created subscribers and are not entitled to receive any messages created prior to the time they are created (that is, messages published before the subscriber re-connects are not resent).

File Locking

Each EMS server writes persistent messages to a store file. To prevent two servers from using the same store file, each server restricts access to its store file for the duration of the server process.

Windows On Windows platforms, servers use the standard Windows `CreateFile` function, supplying `FILE_SHARE_READ` as the `dwShareMode` (third parameter position) to restrict access to other servers.

UNIX On UNIX platforms, servers use the standard `fcntl` operating system call to implement cooperative file locking:

```
struct flock fl;
int err;

fl.l_type = F_WRLCK;
fl.l_whence = 0;
fl.l_start = 0;
fl.l_len = 0;

err = fcntl(file, F_SETLK, &fl);
```

To ensure correct locking, we recommend checking the operating system documentation for this call, since UNIX variants differ in their implementations.

Character Encoding in Messages

Character encodings are named sets of numeric values for representing characters. For example, ISO 8859-1, also known as Latin-1, is the character encoding containing the letters and symbols used by most Western European languages. If your applications are sending and receiving messages that use only English language characters (that is, the ASCII character set), you do not need alter your programs to handle different character encodings. The TIBCO Enterprise Message Service server and application APIs automatically handle ASCII characters in messages.

Character sets become important when your application is handling messages that use non-ASCII characters (such as Japanese language). Also, clients encode messages by default as UTF-8. Some character encodings use only one byte to represent each character, but UTF-8 can potentially use two bytes to represent the same character. For example, the Latin-1 is a single-byte character encoding. If all strings in your messages contain only characters that appear in the Latin-1 encoding, you can potentially improve performance by specifying Latin-1 as the encoding for strings in the message.

TIBCO Enterprise Message Service clients can specify a variety of common character encodings for strings in messages. The character encoding for a message applies to strings that appear in any of the following places within a message:

- property names and property values
- MapMessage field names and values
- data within the message body

The EMS client APIs (Java, .NET and C) include mechanisms for handling strings and specifying the character encoding used for all strings within a message. The following sections describe the implications of string character encoding for TIBCO Enterprise Message Service clients.



Nearly all character sets include unprintable characters. EMS software does not prevent programs from using unprintable characters. However, messages containing unprintable characters (whether in headers or data) can cause unpredictable results if you instruct EMS to print them. For example, if you enable the message tracing feature, EMS prints messages to a trace log file.

Supported Character Encodings

Each message contains the name of the character encoding used to encode strings within the message. This character encoding name is one of the canonical names for character encodings contained in the Java specification. You can obtain a list of canonical character encoding names from the following location:

<http://java.sun.com/j2se/1.4/docs/guide/intl/encoding.doc.html>

Java and .NET clients use these canonical character encoding names when setting or retrieving the character encoding names. C clients have a list of macros that correspond to these canonical names. See the C API references for a list of supported character encodings in these interfaces.

Sending Messages

When a client sends a message, the message stores the character encoding name used for strings in that message. Java clients represent strings using Unicode. A message created by a Java client that does not specify an encoding will use UTF-8 as the named encoding within the message. UTF-8 uses up to four bytes to represent each character, so a Java client can improve performance by explicitly using a single-byte character encoding, if possible.

Java clients can globally set the encoding to use with the `setEncoding` method or the client can set the encoding for each message with the `setMessageEncoding` method. For more information about these methods, see the *TIBCO Enterprise Message Service Java API Reference*.

Typically, C clients manipulate strings using the character encoding of the machine on which they are running. TIBCO Enterprise Message Service provides a character encoding library for C clients to determine the encoding in messages and convert strings to and from Unicode. C clients should explicitly set the character encoding they are using when they create and send a message. For more information, see *TIBCO Enterprise Message Service C & COBOL API Reference*.

Figure 9 illustrates TIBCO Enterprise Message Service clients sending messages encoded in UTF-8. Java clients use this encoding by default. C clients must explicitly set this encoding and convert strings from the local encoding to UTF-8 before sending the message.

Figure 9 Clients sending UTF-8 encoded messages

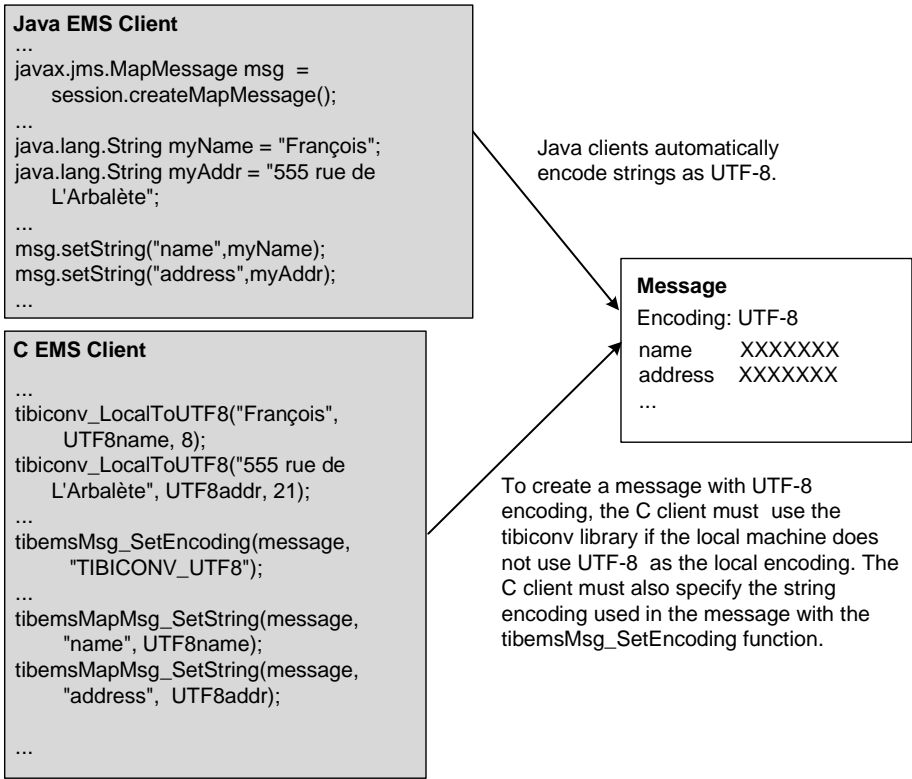
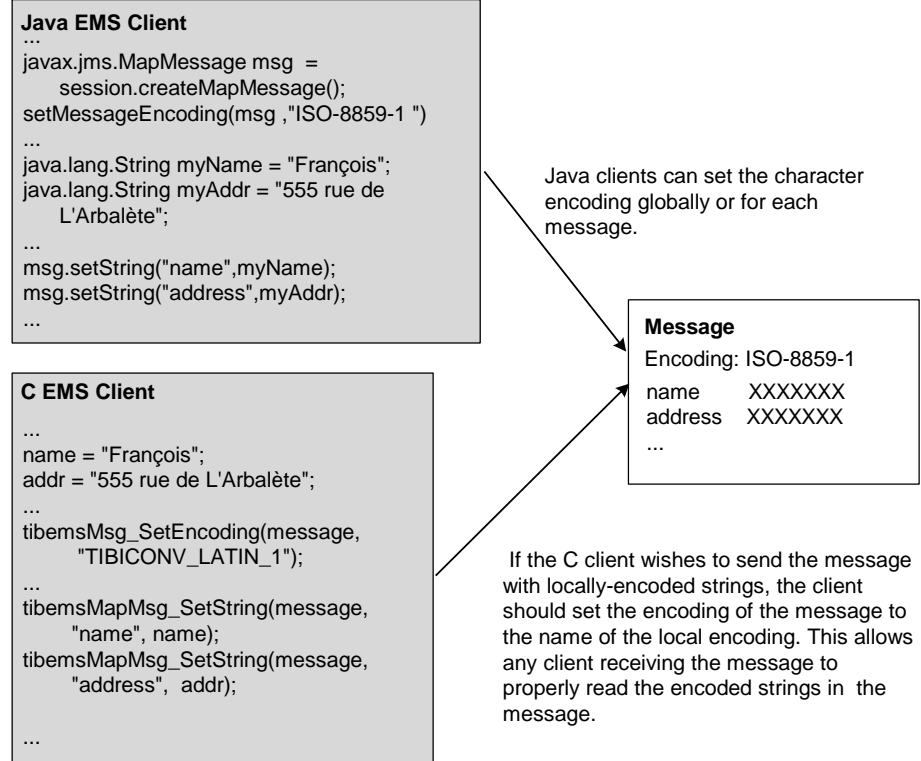


Figure 10 illustrates clients explicitly setting the encoding of strings within a message to ISO-8859-1 (Latin-1). The client must set this encoding explicitly for the message, but there is no need to convert the strings — this happens automatically. The C client's local encoding is Latin-1, so there is no need to convert the strings. However, the C client must specify the encoding of the message before sending.

Figure 10 Clients sending messages with a specific encoding



Receiving Messages

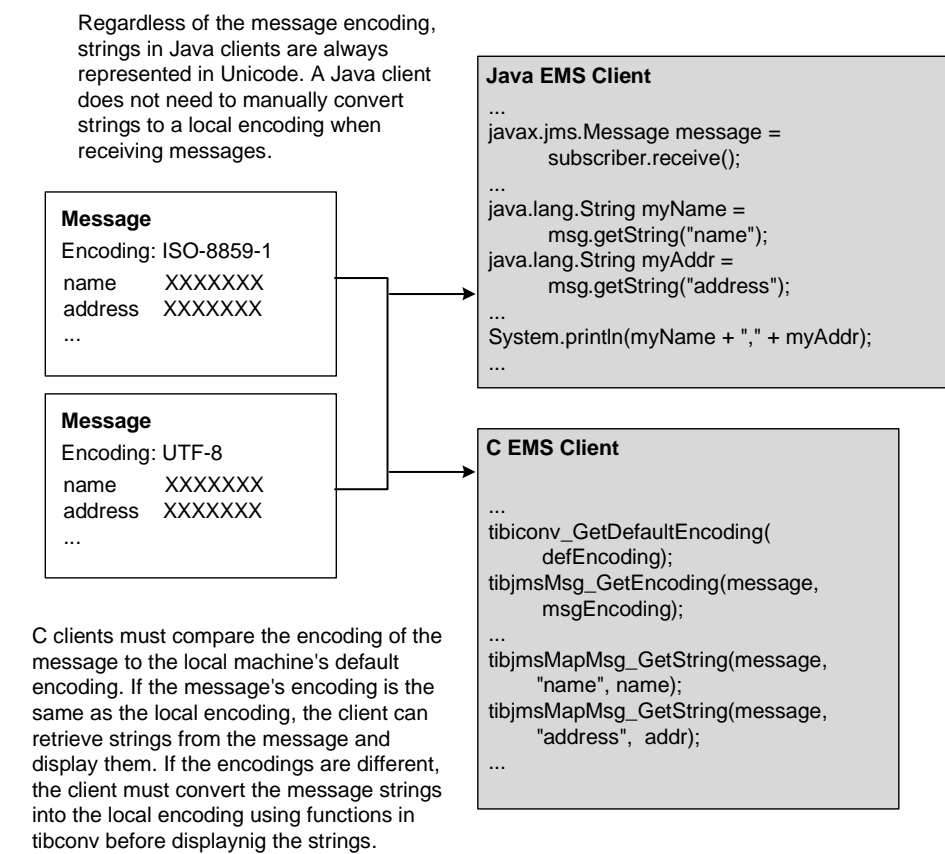
Each message stores the name of the character encoding the sender used. A message receiver can use this information to decode the strings in the message, if necessary.

Java automatically performs any necessary conversion and represents strings in Unicode. Java clients do not need to explicitly perform any operations to display strings stored in a message.

C clients must compare the encoding used for the message with the encoding of the local machine. If the encodings match, the C client can display the string without conversion. If the encodings do not match, the C client must use the `tibconv` library functions to convert the string to the local encoding before the string can be displayed.

Figure 11 illustrates TIBCO Enterprise Message Service clients receiving messages. The Java client can receive the message and display the strings without any additional conversion. The C client must determine the encoding in the message, compare it to the encoding used on the local machine, and then perform conversion, if the encodings do not match.

Figure 11 Clients receiving messages



Message Compression

TIBCO Enterprise Message Service allows the message body to be compressed by the client before the message is sent to the TIBCO Enterprise Message Service server.

About Message Compression

Message compression is especially useful when messages will be stored on the server (persistent queue messages, or topics with durable subscribers). Setting compression ensures that messages will take less memory space in storage.

When messages are compressed and then stored, they are handled by the server in the compressed form. Compression assures that the messages will usually consume less space on disk and will be handled faster by the EMS server.

The compression option only compresses the body of a message. Headers and properties are never compressed. It is best to use compression when the message bodies will be large and the messages will be stored on a server.

When messages will not be stored, compression is not as useful. Compression normally takes time, and therefore the time to send or publish and receive compressed messages is generally longer than the time to send the same messages uncompressed. There is little purpose to message compression for small messages that are not be stored by the server.

Setting Message Compression

Message compression is specified for individual messages. That is, message compression, if desired, is set at the message level. TIBCO Enterprise Message Service does not define a way to set message compression at the per-topic or per-queue level.

To set message compression, the application that sends or publishes the message must access the message properties and set the boolean property `JMS_TIBCO_COMPRESS` to `true` before sending or publishing the message. For example:

```
message.setBooleanProperty("JMS_TIBCO_COMPRESS", true);
```

Compressed messages are handled transparently. The client code only sets the `JMS_TIBCO_COMPRESS` property. The client code does not need to take any other action.

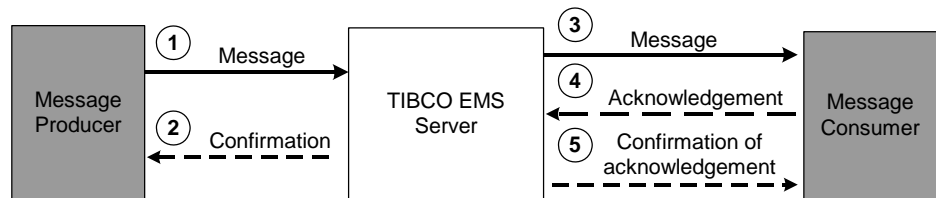
Message Acknowledgement

The interface specification for JMS requires that message delivery be guaranteed under many, but not all, circumstances. The specification defines three levels of acknowledgement:

- `DUPS_OK_ACKNOWLEDGE`, for consumers that are tolerant of duplicate messages.
- `AUTO_ACKNOWLEDGE`, in which the session automatically acknowledges a client's receipt of a message.
- `CLIENT_ACKNOWLEDGE`, in which the client acknowledges the message by calling the message's `acknowledge` method.

Figure 12 illustrates the basic structure of message delivery and acknowledgement.

Figure 12 Message Delivery and Acknowledgement



The following describes the steps in message delivery and acknowledgement:

1. A message is sent from the message producer to the machine on which the TIBCO Enterprise Message Service server resides.
2. The EMS server acknowledges that the message was received.
(Reliable delivery mode omits this acknowledgement to improve performance. However, when the server denies the producer permission to send the message, the server responds to indicate that denial.)
3. The server sends the message to the consumer.
4. The consumer sends acknowledgement to the server that the message was received.
5. In many cases, the server then confirms acknowledgement to the consumer. Acknowledgement from the consumer to the server prevents the delivery of duplicate messages.

Undelivered Message Queue

If a message is to be removed from the system in any way besides being properly consumed by a message consumer, the server checks the message for the `JMS_TIBCO_PRESERVE_UNDELIVERED` property.

When `JMS_TIBCO_PRESERVE_UNDELIVERED` is set to true, the server moves the message to the undelivered message queue, `$sys.undelivered`.

This undelivered message queue is a system queue that is always present and can not be deleted.

To set use of the undelivered message queue, the application that sends or publishes the message must set the boolean `JMS_TIBCO_PRESERVE_UNDELIVERED` property to true before sending or publishing the message. For example:

```
message.setBooleanProperty("JMS_TIBCO_PRESERVE_UNDELIVERED", true);
```

You can only set the undelivered property on individual messages, there is no way to set the undelivered message queue as an option at the per-topic or per-queue level.

You should create a queue receiver to receive and handle messages as they arrive on the undelivered message queue. If you wish to remove messages from the undelivered message queue without receiving them, you can purge the `$sys.undelivered` queue with the administration tool, using the **purge queue** command described under Command Listing on page 167. You can also remove the message using the administrative API included with TIBCO Enterprise Message Service.

Including the Message Sender

Within a message, TIBCO Enterprise Message Service can supply the user name given by the message producer when a connection is created. The `sender_name` and `sender_name_enforced` properties on the destination determine whether the message producer's user name is included in the sent message.

When a user name is included in a message, a message consumer can retrieve that user name by getting the string message property named `JMS_TIBCO_SENDER`. The following illustrates retrieving the property:

```
userID = message.getStringProperty("JMS_TIBCO_SENDER");
```

When the `sender_name` property is enabled and the `sender_name_enforced` property is not enabled on a destination, message producers can specify that the user name is to be left out of the message. Message producers can specify the `JMS_TIBCO_DISABLE_SENDER` boolean property for a particular message, and the message producer's user name will not be included in the message. However, if the `sender_name_enforced` property is enabled, the `JMS_TIBCO_DISABLE_SENDER` property is ignored and the user name is always included in the message.

The following illustrates setting the `JMS_TIBCO_DISABLE_SENDER` property:

```
message.setBooleanProperty("JMS_TIBCO_DISABLE_SENDER", true);
```


Message Extensions

TIBCO Enterprise Message Service extends the `MapMessage` and `StreamMessage` body types. 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 `MapMessage` and `StreamMessage`.

TIBCO Enterprise Message Service adds these two extensions to the `MapMessage` and `StreamMessage` body types:

- 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 `MapMessage` and `StreamMessage` 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 compatibility with Rendezvous messages, see *Message Body* on page 87.

EMS Message Delivery Mode Extensions

TIBCO Enterprise Message Service introduces two types of message delivery which are extensions to the JMS specification.

JMS delivery requirements ensure delivery in almost all circumstances, even if the message receiver is off-line for some time. However, this ensured delivery has a price. This type of delivery requires:

- Two-way network traffic (message and a return message concerning the receipt of the message) for each message or committed transaction of a group of messages.
- Memory allocated for message storage for each persistent message and durable subscriber.

For higher throughput, you may choose one or both of the extensions provided by TIBCO Enterprise Message Service. You might choose this especially if the content of the messages is time-dependent data, such as a stock price quotation.

TIBCO Enterprise Message Service has two extensions to the JMS specification:

- Reliable Message Delivery.
- No-Acknowledgement Message Receipt.

Reliable Message Delivery

JMS has `PERSISTENT` and `NON_PERSISTENT` delivery modes for both topic and queue. EMS extends the set of delivery modes to include reliable delivery.

With `PERSISTENT` delivery, the JMS specification requires the server to return a system message to the producer client application. With `RELIABLE_DELIVERY` mode, the server does not send this system message, and the producer client program does not wait for it. Reliable mode decreases the volume of message traffic, allowing better usage of system resources, and higher message rates.

You can set the delivery mode to reliable in one of two ways:

- Use a `publish()` or `send()` method that accepts a `javax.jms.DeliveryMode` as a parameter.
- Set the delivery mode for the message producer using the following expression:

```
messageProducer.setDeliveryMode(
    com.tibco.tibjms.Tibjms.RELIABLE_DELIVERY);
```



Delivery mode cannot be set by using the `Message.setJMSDeliveryMode()` method. According to the JMS specification, the publisher ignores the value of the `JMSDeliveryMode` header field when a message is being published.

When you use the reliable delivery mode, the client application does not receive any response from the server. Therefore, all publish calls will always succeed (not throw an exception) unless the connection to the server has been terminated.

In some cases a message published in reliable mode may be disqualified and not handled by the server because the destination is not valid or access has been denied. In this case, the message is not sent to any message consumer. However, unless the connection to the server has been terminated, the publishing application will not receive any exceptions, despite the fact that no consumer received the message.

No-Acknowledgement Message Receipt

TIBCO Enterprise Message Service provides a mechanism for not acknowledging the receipt of messages.

In no-acknowledge receipt mode, after the server sends a message to the client, all information regarding that message for that consumer is eliminated from the server. Therefore, there is no need for the client application to send an acknowledgement to the server about the received message. Not sending acknowledgements decreases the message traffic and saves time for the receiver, therefore allowing better utilization of system resources.

No-acknowledgement receipt mode is configured at the session level. Add the `com.tibco.tibjms.Tibjms.NO_ACKNOWLEDGE` constant when you create the session. For example:

```
javax.jms.TopicSession session =
    topicConnection.createTopicSession(
        false, com.tibco.tibjms.Tibjms.NO_ACKNOWLEDGE);
```



Sessions created in no-acknowledge receipt mode cannot be used to create durable subscribers.

Also, queue receivers on a queue that is routed from another server are not permitted to specify `NO_ACKNOWLEDGE` mode.

Chapter 5

Working With TIBCO Rendezvous

This chapter describes the interoperation of TIBCO Enterprise Message Service and TIBCO Rendezvous.

Topics

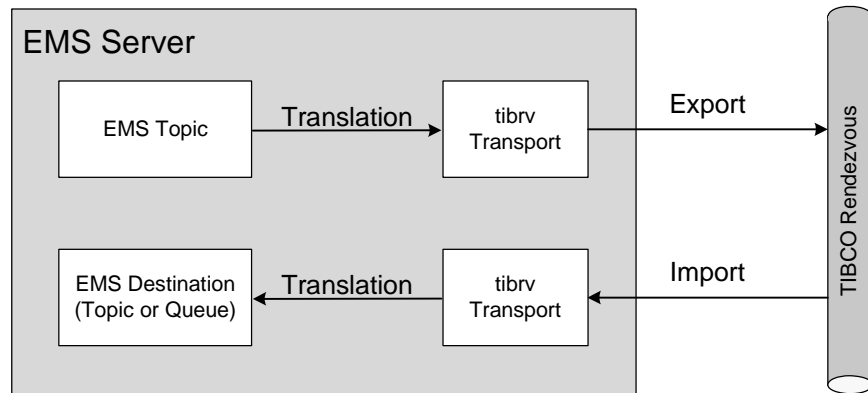
- *Overview, page 74*
- *Configuring Transports for Rendezvous, page 76*
- *Topics, page 79*
- *Queues, page 81*
- *Import Issues, page 83*
- *Export Issues, page 84*
- *Message Translation, page 85*
- *Pure Java Rendezvous Programs, page 91*

Overview

TIBCO Enterprise Message Service (release 4 and later) can exchange messages with TIBCO Rendezvous (release 6.9 and later).

- Scope
- EMS can import and export messages to an external system through an EMS *topic*.
 - EMS can import messages from an external system to an EMS *queue* (but queues cannot export).

Figure 13 Rendezvous Transports in the EMS Server



Message Translation

EMS and Rendezvous use different formats for messages and their data. When `tibemspd` imports or exports a messages, it translates the message and its data to the appropriate format; for details, see Message Translation on page 111.

Configuration

`tibemspd` uses definitions and parameters in four configuration files to guide the exchange of messages with Rendezvous.

- Enabling The parameter `tibrv_transports` (in the configuration file `tibemspd.conf`) globally enables or disables message exchange with Rendezvous. The default value is `disabled`. To use these transports, you must explicitly set this parameter to `enabled`.

Transports	Transport definitions (in the configuration file <code>transports.conf</code>) specify the communication protocol between EMS and the external system; for details, see Configuring Transports for Rendezvous on page 76.
Destinations	<p>Destination definitions (in the configuration files <code>topics.conf</code> and <code>queues.conf</code>) can set the <code>import</code> and <code>export</code> properties to specify one or more transports:</p> <ul style="list-style-type: none"> • <code>import</code> instructs <code>tibemsd</code> to import messages that arrive on those transports from Rendezvous, and deliver them to the EMS destination. • <code>export</code> instructs <code>tibemsd</code> to take messages that arrive on the EMS destination, and export them to Rendezvous via those transports. <p>For details, see Topics on page 104, and Queues on page 106.</p>
RVCN Listeners	<p>When exporting messages on a transport configured for certified message delivery, you can pre-register RVCN listeners in the file <code>tibrvcn.conf</code>.</p> <p>For details, see <code>tibrvcn</code> on page 157, and Certified Messages on page 84</p>

Deprecated Configuration Properties



Previous releases of TIBCO Enterprise Message Service provided properties in `tibemsd.conf` and destination properties for configuring communication with TIBCO Rendezvous. These properties are supported for backward-compatibility, but they are deprecated. This configuration method may not be supported in future releases of TIBCO Enterprise Message Service. You can achieve the same functionality that these parameters provide by using `transports.conf` and the `export` and `import` properties on destinations described in this chapter.

You may continue to use the deprecated properties with this release, but you should switch to using the new approach as soon as possible. For lists of deprecated parameters and properties, see [TIBCO Rendezvous Parameters—Deprecated](#) on page 146, and [Deprecated Properties](#) on page 34.

Configuring Transports for Rendezvous

Transports mediate the flow of messages between TIBCO Enterprise Message Service and TIBCO Rendezvous.

`timemsd` connects to Rendezvous daemons in the same way as any other Rendezvous client would. Transport definitions (in the file `transports.conf`) configure the behavior of these connections. You must properly configure these transports.

Transport Definitions

`transports.conf` contains zero or more transport definitions. Each definition begins with the name of a transport, surrounded by square brackets. Subsequent lines set the parameters of the transport.

Table 8 Rendezvous: Transport Parameters (Sheet 1 of 3)

Parameter	Description
<code>type</code>	Required. For Rendezvous transports, the value must be either <code>tibrv</code> or <code>tibrvc</code> .
Rendezvous Parameters	
The syntax and semantics of these parameters are identical to the corresponding parameters in Rendezvous clients. For full details, see the Rendezvous documentation set.	
<code>service</code>	When absent, the default value is 7500.
<code>network</code>	When absent, the default value is the host computer's primary network.
<code>daemon</code>	When absent, the default value is an <code>rvc</code> process on the local host computer. To connect to a non-default daemon, supply <code>hostname:protocol:port</code> . You may omit any of the three parts. The default <code>hostname</code> is the local host computer. The default protocol is <code>tcp</code> . The default <code>port</code> is 7500.
Rendezvous Certified Messaging (RVCM) Parameters	
Use these properties only for <code>tibrvc</code> transports.	
The syntax and semantics of these parameters are identical to the corresponding parameters in Rendezvous CM clients. For full details, see the Rendezvous documentation set.	
<code>cm_name</code>	Correspondent name.

Table 8 Rendezvous: Transport Parameters (Sheet 2 of 3)

Parameter	Description
<code>rv_tport</code>	Required. Each RVCN transport depends in turn upon an ordinary Rendezvous transport. Set this parameter to the name of a Rendezvous transport (type <code>tibrv</code>) defined in the EMS configuration file <code>transports.conf</code> .
<code>ledger_file</code>	Name for file-based ledger.
<code>sync_ledger</code>	<code>true</code> or <code>false</code> . If <code>true</code> , operations that update the ledger do not return until changes are written to the storage medium.
<code>request_old</code>	<code>true</code> or <code>false</code> . If <code>true</code> , this transport server requests unacknowledged messages sent from other RVCN senders while this transport was unavailable.
<code>default_ttl</code>	This parameter sets default CM time limit (in seconds) for all CM messages exported on this transport.
<code>explicit_config_only</code>	<p><code>true</code> or <code>false</code>. If <code>true</code>, <code>tibemsd</code> allows RVCN listeners to register for certified delivery only if they are configured in advance with the EMS server (either in <code>tibrvcn.conf</code> or using the <code>create rvcnlistener</code> command). That is, <code>tibemsd</code> ignores registration requests from non-configured listeners.</p> <p>If <code>false</code> (the default), <code>tibemsd</code> allows any RVCN listener to register.</p>
EMS Parameters	
<code>topic_import_dm</code> <code>queue_import_dm</code>	<p>EMS sending clients can set the <code>JMSDeliveryMode</code> header field for each message. However, Rendezvous clients cannot set this header. Instead, these two parameters determine the delivery modes for all topic messages and queue messages that <code>tibemsd</code> imports on this transport.</p> <p><code>TIBEMS_PERSISTENT</code> <code>TIBEMS_NON_PERSISTENT</code> <code>TIBEMS_RELIABLE</code></p> <p>When absent, the default is <code>TIBEMS_NON_PERSISTENT</code>.</p>
<code>export_headers</code>	<p>When <code>true</code>, <code>tibemsd</code> includes JMS header fields in exported messages.</p> <p>When <code>false</code>, <code>tibemsd</code> suppresses JMS header fields in exported messages.</p> <p>When absent, the default value is <code>true</code>.</p>

Table 8 Rendezvous: Transport Parameters (Sheet 3 of 3)

Parameter	Description
export_properties	When true, tibemsd includes JMS properties in exported messages. When false, tibemsd suppresses JMS properties in exported messages. When absent, the default value is true.

Example

These examples from transports.conf illustrate the syntax of transport definitions.

```
[RV01]
type = tibrv
topic_import_dm = TIBEMS_RELIABLE
queue_import_dm = TIBEMS_PERSISTENT
service = 7780
network = lan0
daemon = tcp:host5:7885

[RV02]
type = tibrv
service = 7890
network = lan0
daemon = tcp:host5:7995

[RVCM01]
type = tibrvcn
export_headers = true
export_properties = true
rv_tport = RV02
cm_name = RVCMTrans1
ledger_file = ledgerFile.store
sync_ledger = true
request_old = true
default_ttl = 600
```

Topics

Topics can both export and import messages. Accordingly, you can configure topic definitions (in the configuration file `topics.conf`) with `import` and `export` properties that specify one or more external transports:

- `import`
 - `import` instructs `tibemsd` to import messages that arrive on those transports from Rendezvous, and deliver them to the EMS destination.
- `export`
 - `export` instructs `tibemsd` to take messages that arrive on the EMS destination, and export them to Rendezvous via those transports.



The EMS server *never* re-exports an imported message on the same topic.

(For general information about `topics.conf` syntax and semantics, see topics on page 150. You can also configure topics using the administration tool command `addprop topic`.)

- Example** For example, the following `tibemsadmin` commands configure the topic `myTopics.news` to import messages on the transports `RV01` and `RV02`, and to export messages on the transport `RV02`.

```
addprop topic myTopics.news import="RV01,RV02"
addprop topic myTopics.news export="RV02"
```

Rendezvous messages with subject `myTopics.news` arrive at `tibemsd` over the transports `RV01` and `RV02`. EMS clients can receive those messages by subscribing to `myTopics.news`.

EMS messages sent to `myTopics.news` are exported to Rendezvous over transport `RV02`. Rendezvous clients of the corresponding daemons can receive those messages by subscribing to `myTopics.news`.

Import Only when Subscribers Exist

When a topic specifies `import` on a connected transport, `tibemsd` imports messages only when the topic has registered subscribers.

Wildcards

Wildcards in the `import` and `export` properties obey EMS syntax and semantics (which is identical to Rendezvous syntax and semantics); see *Destination Name—Syntax and Semantics* on page 102.

Certified Messages

You can import and export TIBCO Rendezvous certified messages (`tibrvc` transport) to EMS topics. Rendezvous certified transports guarantee message delivery.

RVCM Ledger `tibrvc` transports can store information about subjects in a ledger file. You can review the ledger file using an administration tool command; see `show rvctransportledger` on page 193).

For more information about ledger files, see TIBCO Rendezvous documentation.

Queues

Queues can import messages, but cannot export them.

Configuration

You can configure queue definitions (in the configuration file `queues.conf`) with the `import` property that specify one or more external transports.

- `import` instructs `tibemsd` to import messages that arrive on those transports from Rendezvous, and deliver them to the EMS destination.

(For general information about `queues.conf` syntax and semantics, see queues on page 150. You can also configure queues using the administration tool command `addprop queue`.)

Example For example, the following `tibemsadmin` command configures the queue `myTopics.news` to import messages on the transports `RV01` and `RV02`.

```
addprop topic myQueue.in import="RV01,RV02"
```

Rendezvous messages with subject `myQueue.in` arrive at `tibemsd` over the transports `SS01` and `SS02`. EMS clients can receive those messages by subscribing to `myQueue.in`.

Import—Start and Stop

When a queue specifies `import` on a connected transport, `tibemsd` immediately begins importing messages to the queue, even when no receivers exist for the queue.

For static queues (configured by an administrator) `tibemsd` continues importing until you explicitly delete the queue.

Wildcards

Wildcards in the `import` property obey EMS syntax and semantics (not Rendezvous syntax and semantics); see Destination Name—Syntax and Semantics on page 102.

EMS clients cannot subscribe to wildcard queues—however, you can define wildcard queues in the EMS server for the purpose of property inheritance. That is, you can configure a static queue named `foo.*` and set properties on it, so that child queues named `foo.bar` and `foo.baz` will both inherit those properties.

If you define a queue that imports `foo.*`, `tibemsd` begins importing all matching messages from Rendezvous. As messages arrive, `tibemsd` creates dynamic child queues (for example, `foo.bar` and `foo.baz`) and delivers the messages to them. Notices that `tibemsd` delivers messages to these dynamic child queues even when no subscribers exist to drain them.

Import Issues

This section presents issues associated with importing messages to EMS from Rendezvous—whether on a topic or a queue.

Import Destination Names Must be Unique



When a topic and a queue share the same name, *at most one* of them may set the `import` property. For example, if a topic `foo.bar` and a queue `foo.bar` are both defined, only one may specify the `import` property.

JMSReplyTo

When `tibemsd` imports and translates a Rendezvous message, it sets the `JMSReplyTo` field of the EMS message to the value of the Rendezvous reply subject, so that EMS clients can reply to the message.

Usually this value represents a Rendezvous subject. You must explicitly configure `tibemsd` to create a topic with a corresponding name, which exports messages to Rendezvous.

Guaranteed Delivery



For full end-to-end certified delivery from Rendezvous to EMS, all three of these conditions must be true:

- Rendezvous senders must send labeled messages on RVCM transports.
- The transport definition must set `topic_import_dm` or `queue_import_dm` (as appropriate) to `TIBEMS_PERSISTENT`.
- A durable subscription for the EMS topic or queue must exist.

Export Issues

This section presents issues associated with exporting messages from EMS to Rendezvous.

JMSReplyTo

Topics	Consider an EMS message in which the field <code>JMSReplyTo</code> contains a topic. When exporting such a message to Rendezvous, you must explicitly configure <code>tibemsd</code> to import replies from Rendezvous to that reply topic.
Temporary Topics	Consider an EMS message in which the field <code>JMSReplyTo</code> contains a temporary topic. When <code>tibemsd</code> exports such a message to Rendezvous, it <i>automatically</i> arranges to import replies to that temporary topic from Rendezvous; you do not need to configure it explicitly.

Certified Messages

RVC Registration	<p>When an RVC listener receives its first labeled message, it registers to receive subsequent messages as certified messages. Until the registration is complete, it receives labeled messages as reliable messages. When exporting messages on a <code>tibrvc</code> transport, we recommend either of two actions to ensure certified delivery for all exported messages:</p> <ul style="list-style-type: none"> • Create the RVC listener before sending any messages from EMS clients. • Pre-register an RVC listener, either with the administration tool (see <code>create rvcmlistener</code> on page 171), or in the configuration file <code>tibrvc.conf</code> (see <code>tibrvc</code> on page 157).
---------------------	--

Guaranteed Delivery



For full end-to-end certified delivery to Rendezvous from EMS, the following condition must be true:

- EMS senders must send persistent messages.

Message Translation

JMS Header Fields

EMS supports the ten predefined JMS header fields; see Header Fields on page 56.

Two Special Cases

These two header fields are special cases:

- JMS header `JMSDestination` corresponds to Rendezvous subject.
- JMS header `JMSReplyTo` corresponds to Rendezvous reply subject.

Import

When importing a Rendezvous message to an EMS message, `tibemsd` does not set any JMS header fields, except for the special cases noted above.

Export

When exporting an EMS message to a Rendezvous message, `tibemsd` groups all the JMS header fields (except for the special cases noted above) into a single submessage within the Rendezvous message. The field `JMSHeaders` contains that submessage. Fields of the submessage map the names of JMS header fields to their values.

`tibemsd` ignores any JMS header fields that are null or absent—it omits them from the exported message.

You can instruct `tibemsd` to suppress the entire header submessage in all exported messages by setting the transport property `export_headers = false`.

Table 9 presents the mapping of JMS header fields to Rendezvous data types (that is, the type of the corresponding field in the exported message).

Table 9 Rendezvous: Mapping JMS Header Fields to RV Datatypes (Sheet 1 of 2)

JMS Header Name	Rendezvous Type
<code>JMSDeliveryMode</code>	<code>TIBRVMSG_U8</code>
<code>JMSPriority</code>	<code>TIBRVMSG_U8</code>
<code>JMSTimestamp</code>	<code>TIBRVMSG_U64</code>
<code>JMSExpiration</code>	<code>TIBRVMSG_U64</code>
<code>JMSType</code>	<code>TIBRVMSG_STRING</code>
<code>JMSMessageID</code>	<code>TIBRVMSG_STRING</code>
<code>JMSCorrelationID</code>	<code>TIBRVMSG_STRING</code>

Table 9 Rendezvous: Mapping JMS Header Fields to RV Datatypes (Sheet 2 of 2)

JMS Header Name	Rendezvous Type
JMSRedelivered	TIBRVMSG_BOOL
JMSDestination	send subject in TIBCO Rendezvous
JMSReplyTo	reply subject in TIBCO Rendezvous

JMS Property Fields

Import When importing a Rendezvous message to an EMS message, `tibemsd` sets these JMS properties:

- `JMS_TIBCO_IMPORTED` gets the value `true`, to indicates that the message did not originate from an EMS client.
- `JMS_TIBCO_MSG_EXT` gets the value `true`, to indicate that the message *might* contain submessage fields or array fields.

Import RVC In addition to the two fields described above, when `tibemsd` imports a certified message on a `tibrvc` transport, it can also set these properties (if the corresponding information is set in the Rendezvous message):

- `JMS_TIBCO_CM_PUBLISHER` gets a string value indicating the correspondent name of the Rendezvous CM transport that sent the message (that is, the sender name).
- `JMS_TIBCO_CM_SEQUENCE` gets a long value indicating the CM sequence number of the message.

Export When exporting an EMS message to a Rendezvous message, `tibemsd` groups all the JMS property fields into a single submessage within the Rendezvous message. The field `JMSProperties` contains that submessage. Fields of the submessage map the names of JMS property fields to their values.

`tibemsd` ignores any JMS property fields that are not set, or are set to null—it omits them from the exported message.

You can instruct `tibemsd` to suppress the entire properties submessage in the exported message by setting the transport property `export_properties = false`.

Message Body

`tibemsd` can export messages with any JMS message body type to TIBCO Rendezvous. Conversely, `tibemsd` can import messages with any message type from TIBCO Rendezvous.

For information about JMS body types, see Message Bodies on page 57.

For information about the structure of messages, see JMS Message Structure on page 56.

Import When importing a Rendezvous message, `tibemsd` translates it to an EMS message body types based on the presence of the fields in Table 10.

Table 10 Rendezvous: Mapping Message Types (Import)

Rendezvous Field	EMS Body Type
JMSBytes	JMSBytesMessage
JMSObject	JMSObjectMessage
JMSStream	JMSStreamMessage
JMSText	JMSTextMessage
None of these fields are present.	JMSMapMessage



The field names `DATA` and `_data_` are reserved. We strongly discourage you from using these field names in either EMS and Rendezvous applications, and especially when these two message transport mechanisms interoperate.

Export When exporting an EMS message, `tibemsd` translates it to a Rendezvous message with the following structure:

- The field `JMSHeaders` contains a submessage; see JMS Header Fields on page 85. When the transport parameter `export_headers` is `false`, this field is omitted.
- The field `JMSProperties` contains a submessage; see JMS Property Fields on page 86. When the transport parameter `export_properties` is `false`, this field is omitted.

- When translating the data fields of an EMS message, the results depend on the JMS body type. Table 11 specifies the mapping.

Table 11 *Rendezvous: Mapping Message Types (Export)*

JMS Body Type	Export Translation
BytesMessage	<p>The message data translates to a byte array that contains the bytes of the original EMS message.</p> <p>The field JMSBytes receives this data. It has type TIBRVMSG_OPAQUE.</p>
ObjectMessage	<p>The message data translates to a byte array containing the serialized Java object.</p> <p>The field JMSObject receives this data. It has type TIBRVMSG_OPAQUE.</p>
StreamMessage	<p>The message data translates to a byte array that encodes the objects in the original EMS message.</p> <p>The field JMSStream receives this data. It has type TIBRVMSG_OPAQUE.</p>
TextMessage	<p>The message data translates to a UTF-8 string corresponding to the text of the original EMS message.</p> <p>The field JMSText receives this data. It has type TIBRVMSG_STRING.</p>
MapMessage	<p>The message data fields map directly to top-level fields in the Rendezvous message. The fields retain the same names as in the original EMS message.</p> <p>See also, Message Extensions on page 69.</p>

Data Types

Table 12 presents the mapping between EMS datatypes and Rendezvous datatypes. The mapping is bidirectional, except for the Rendezvous types that have no corresponding EMS type (for these types the mapping is marked as unidirectional in the middle column of Table 12).

Table 12 *Rendezvous: Mapping Data Types (Sheet 1 of 3)*

EMS	Map	Rendezvous
Boolean		TIBRVMSG_BOOL
Byte		TIBRVMSG_I8
Short	<—	TIBRVMSG_U8

Table 12 Rendezvous: Mapping Data Types (Sheet 2 of 3)

EMS	Map	Rendezvous
Short		TIBRVMSG_I16
Integer	<—	TIBRVMSG_U16
Integer		TIBRVMSG_I32
Long	<—	TIBRVMSG_U32
Long		TIBRVMSG_I64
Long	<—	TIBRVMSG_U64
Float		TIBRVMSG_F32
Double		TIBRVMSG_F64
Short	<—	TIBRVMSG_IPPORT16
Integer	<—	TIBRVMSG_IPADDR32
MapMessage		TIBRVMSG_MSG
Long	<—	TIBRVMSG_DATETIME
byte[]		TIBRVMSG_OPAQUE
java.lang.String		TIBRVMSG_STRING
byte[]	<—	TIBRVMSG_XML
byte[]	<—	TIBRVMSG_I8ARRAY
short[]	<—	TIBRVMSG_U8ARRAY
short[]		TIBRVMSG_I16ARRAY
int[]	<—	TIBRVMSG_U16ARRAY
int[]		TIBRVMSG_I32ARRAY
long[]	<—	TIBRVMSG_U32ARRAY
long[]		TIBRVMSG_I64ARRAY
long[]	<—	TIBRVMSG_U64ARRAY

Table 12 Rendezvous: Mapping Data Types (Sheet 3 of 3)

EMS	Map	Rendezvous
float[]		TIBRVMSG_F32ARRAY
double[]		TIBRVMSG_F64ARRAY

Pure Java Rendezvous Programs

TIBCO Enterprise Message Service is shipped with the `tibrvjms.jar` file that you can include in your TIBCO Rendezvous applications. This JAR file includes the implementation of the `com.tibco.tibrv.TibrvJMSTransport` class. This class extends the `com.tibco.tibrv.TibrvNetTransport` class and allows your pure Java Rendezvous programs to communicate directly with the EMS server instead of through `rva`.

To use the `TibrvJMSTransport` class, your application must include `tibrvjms.jar` (included with TIBCO Enterprise Message Service) and `tibrvjweb.jar` (included with TIBCO Rendezvous).



You can use `TibrvJMSTransport` only in Rendezvous applications. This class is not intended for use in your EMS Java clients.

Both TIBCO Rendezvous and TIBCO Enterprise Message Service must be purchased, installed, and configured before creating pure Java Rendezvous applications that use the `TibrvJMSTransport` class.

The `TibrvJMSTransport` class provides Rendezvous reliable communication only. Other types of communication, such as certified messaging, are not supported by this transport.

Applications using this transport can send messages to a topic on an EMS server that has the same topic name as the subject of the message. EMS topics receiving Rendezvous messages sent by way of the `TibrvJMSTransport` do not need to specify the `import` property. This transport cannot be used to send messages to JMS queues.

For more information about `TibrvNetTransport` and how to create use transports in TIBCO Rendezvous Java programs, see TIBCO Rendezvous documentation. Table 13 on page 92 describes the additional methods of `TibrvJMSTransport`.

Table 13 *TibrvJMSTransport class (Sheet 1 of 2)*

Method	Description
<code>TibrvJMSTransport()</code> throws <code>TibrvException</code>	Constructor for creating a <code>TibrvJMSTransport</code> .
<code>TibrvJMSTransport(String serverURL)</code> throws <code>TibrvException</code>	If no parameters are passed, the transport assumes the server is running on the same machine with the default listener port.
<code>TibrvJMSTransport(String serverURL, String clientId, String userName, String password)</code> throws <code>TibrvException</code>	You can also pass the <code>serverURL</code> to connect to the TIBCO Enterprise Message Service server at the specified location.
<code>TibrvJMSTransport(String serverURL, String clientId, String userName, String password, Hashtable sslParameters, boolean emulateReconnect)</code> throws <code>TibrvException</code>	<p>You may also pass the <code>clientId</code> of your client connection, and a username and a password to use to connect to the server.</p> <p>SSL parameters govern the connection between the Rendezvous Java client and <code>tibemsd</code>.</p> <p>When <code>emulateReconnect</code> is <code>true</code>, the first argument can be a list of server URLs, and the <code>recoverConnection</code> method attempts to reconnect to one of them.</p>
<code>destroy()</code>	Destroys the transport and all associated listeners.
<code>toString()</code>	Returns a string representation of this transport.
<code>setPersistentDelivery(boolean persistent)</code>	Sets the TIBCO Enterprise Message Service persistent delivery mode for messages sent or received on this transport.
<code>isPersistentDelivery()</code>	Returns <code>true</code> if persistent delivery mode is set for this transport.

Table 13 TibrvJMSTransport class (Sheet 2 of 2)

Method	Description
<code>recoverConnection()</code> throws <code>TibrvException</code>	Reconnect to a server. Try servers in sequence, starting with the most recently connected server. If the call cannot connect to any server, it throws an exception.
<code>getCurrentConnectedServer()</code>	Return a string identifying the server to which the transport is currently connected.

Chapter 6

Working With TIBCO SmartSockets

This chapter describes the interoperation of TIBCO Enterprise Message Service and TIBCO SmartSockets.

Topics

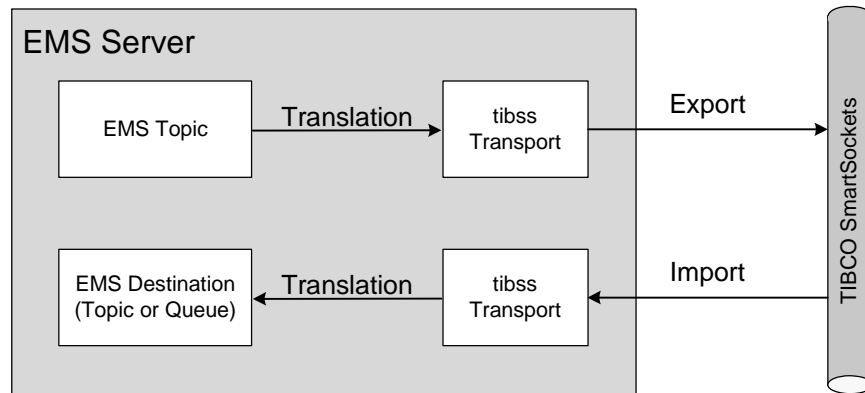
- *Overview, page 96*
- *Configuring Transports for SmartSockets, page 98*
- *Topics, page 104*
- *Queues, page 106*
- *Import Issues, page 108*
- *Export Issues, page 109*
- *Message Translation, page 111*

Overview

TIBCO Enterprise Message Service (release 4 and later) can exchange messages with TIBCO SmartSockets (release 6.5 and later).

- Scope
- EMS can import and export messages to an external system through an EMS *topic*.
 - EMS can import messages from an external system to an EMS *queue* (but queues cannot export).

Figure 14 SmartSockets Transports in the EMS Server



Message Translation

EMS and SmartSockets use different formats for messages and their data. When `tibemspd` imports or exports a messages, it translates the message and its data to the appropriate format; for details, see Message Translation on page 111.

Configuration

`tibemspd` uses definitions and parameters in three configuration files to guide the exchange of messages with SmartSockets.

- Enabling The parameter `tibss_transports` (in the configuration file `tibemspd.conf`) globally enables or disables message exchange with SmartSockets. The default value is `disabled`. To use these transports, you must explicitly set this parameter to `enabled`.

The parameter `tibss_config_dir` (in the configuration file `tibemsd.conf`) specifies the location of SmartSockets files needed by the SmartSockets client within `tibemsd`.

Transports Transport definitions (in the configuration file `transports.conf`) specify the communication protocol between EMS and the external system; for details, see [Configuring Transports for SmartSockets](#) on page 98.

Destinations Destination definitions (in the configuration files `topics.conf` and `queues.conf`) can set the `import` and `export` properties to specify one or more transports:

- `import` instructs `tibemsd` to import messages that arrive on those transports from SmartSockets, and deliver them to the EMS destination.
- `export` instructs `tibemsd` to take messages that arrive on the EMS destination, and export them to SmartSockets via those transports.

For details, see [Topics](#) on page 104, and [Queues](#) on page 106.

Starting the Servers

We recommend starting the SmartSockets RTserver before starting `tibemsd`.

Configuring Transports for SmartSockets

Transports mediate the flow of messages between TIBCO Enterprise Message Service and TIBCO SmartSockets.

`timemsd` connects to SmartSockets RTservers in the same way as any other SmartSockets client. Transport definitions (in the file `transports.conf`) configure the behavior of these connections. You must properly configure these transports.

Transport Definitions

`transports.conf` contains zero or more transport definitions. Each definition begins with the name of a transport, surrounded by square brackets. Subsequent lines set the parameters of the transport.

Table 14 SmartSockets: Transport Parameters (Sheet 1 of 4)

Parameter	Description
<code>type</code>	Required. For SmartSockets transports, the value must be <code>tibss</code> .
SmartSockets Parameters	
The syntax and semantics of these parameters are identical to the corresponding parameters in SmartSockets clients. For full details, see the SmartSockets documentation set.	
<code>server_names</code>	<p>The value is a comma-separated list specifying connections to one or more SmartSockets RTservers.</p> <p>Each item in the list has the form <i>protocol:hostname:port</i>. You may omit any of the three parts. The default <i>hostname</i> is the local host computer. The default protocols and ports vary with hardware and operating system platforms; on Windows platforms, the default protocol is <code>tcp</code> and the default <i>port</i> is 5101.</p> <p>A list of several servers specifies fault tolerance—<code>timemsd</code> attempts to connect to them in the order listed.</p> <p>When this parameter is absent, the default instructs the EMS server to attempt to connect to an RTserver on the local host computer (the same computer as the EMS server), using default protocols and ports.</p>
<code>username</code> <code>password</code>	<code>timemsd</code> uses these two parameters to authenticate itself to the SmartSockets servers.

Table 14 SmartSockets: Transport Parameters (Sheet 2 of 4)

Parameter	Description
<code>project</code>	SmartSockets uses projects to maintain orthogonal subject name-spaces. When absent, the default project is <code>rtworks</code> .
<code>delivery_mode</code>	This parameter determines the quality of service with which delivers messages to the SmartSockets server over this transport: <code>best_effort</code> <code>gmd_all</code> <code>gmd_some</code> <code>ordered</code> When absent, the default is <code>best_effort</code> .
<code>lb_mode</code>	SmartSockets servers balance the message load by distributing messages among several clients. This parameter determines the load balancing regimen for messages that this transport exports to the SmartSockets server. <code>none</code> <code>round_robin</code> <code>weighted</code> <code>sorted</code> When absent, the default is <code>none</code> .
<code>override_lb_mode</code>	<code>enable</code> instructs the RTserver to deliver all messages on this client connection—even if other clients participate in load balancing. For example, even though many order-processing clients might share the load of order messages, a message logging facility would require all order messages, rather than a subset. <code>disable</code> informs the RTserver that this client (that is, the EMS server) participates in load balancing (for example, sharing the load with other EMS servers). When absent, the default is <code>enable</code> .
<code>gmd_file_delete</code>	SmartSockets clients keep data for guaranteed message delivery (GMD) in a store file. <code>disable</code> instructs <code>tibemsd</code> to open the existing GMD store file. <code>enable</code> instructs <code>tibemsd</code> to delete the GMD store file and create a new one when creating this transport. When absent, the default is <code>disable</code> .

Table 14 SmartSockets: Transport Parameters (Sheet 3 of 4)

Parameter	Description
EMS Parameters	
topic_import_dm queue_import_dm	<p>EMS sending clients can set the JMSDeliveryMode header field for each message. However, SmartSockets clients cannot set this header. Instead, these two parameters determine the delivery modes for all topic messages and queue messages that tibemsd imports on this transport.</p> <p>TIBEMS_PERSISTENT TIBEMS_NON_PERSISTENT TIBEMS_RELIABLE</p> <p>When absent, the default is TIBEMS_NON_PERSISTENT.</p>
export_headers	<p>When true, tibemsd includes JMS header fields in exported messages.</p> <p>When false, tibemsd suppresses JMS header fields in exported messages.</p> <p>When absent, the default value is true.</p>
export_properties	<p>When true, tibemsd includes JMS properties in exported messages.</p> <p>When false, tibemsd suppresses JMS properties in exported messages.</p> <p>When absent, the default value is true.</p>
import_ss_headers	<p>This parameter governs the import of SmartSockets message headers to EMS properties.</p> <p>The value can be none, type_num, or all. For complete details, see SmartSockets Message Properties on page 112.</p> <p>When absent, the default value is none.</p>
subscribe_mode	<ul style="list-style-type: none">When subscriptions do <i>not</i> collide, specify exact, for best performance.When subscriptions collide, specify all, for correct semantics. <p>When absent, the default is exact.</p> <p>For a definition of wildcard collision, and complete details about the operation of these two modes, see Subscribe Mode on page 102.</p>

Table 14 SmartSockets: Transport Parameters (Sheet 4 of 4)

Parameter	Description
preserve_gmd	<p>This parameter determines the behavior of the EMS server when it has exported a GMD message to SmartSockets, and SmartSockets cannot deliver that message. When SmartSockets returns the undelivered message, EMS can either preserve it in the EMS undelivered message queue, or discard it.</p> <ul style="list-style-type: none">• <code>always</code> instructs EMS to preserve all undelivered GMD messages in the EMS undelivered message queue.• <code>receivers</code> instructs EMS to preserve only those undelivered GMD messages that SmartSockets could not deliver despite the existence of one or more GMD receivers. That is, if SmartSockets cannot deliver a message because no GMD receivers exist, then EMS does not preserve the undelivered message.• <code>never</code> instructs EMS to discard all undelivered SmartSockets GMD messages. <p>When absent, the default value is <code>never</code>.</p> <p>This parameter applies only when the transport's <code>delivery_mode</code> parameter is either <code>gmd_all</code> or <code>gmd_some</code>.</p> <p>When the EMS server preserves a GMD message, it follows these rules to convert the returned SmartSockets message to an EMS message:</p> <ul style="list-style-type: none">• Follow all general rules for importing messages; see Message Translation on page 111.• Disregard the value of the <code>import_ss_headers</code> parameter, and instead import all SmartSockets headers (as if the value of <code>import_ss_headers</code> were <code>all</code>). For a list of headers, see SmartSockets Message Properties on page 112.• Set the value of <code>JMS_TIBCO_SS_EXPIRATION</code> to the current time—that is, the time at which the SmartSockets server returned the undelivered message to EMS. (Notice that the <code>this</code> header would otherwise remain unused, since GMD messages do not expire.)

Example

These examples from `transports.conf` illustrate the syntax of transport definitions.

```
[SS01]
type = tibss
```

```

server_names = rtHost1
username = emsServer6
password = myPasswd
project = sales_order_entry

[SS02]
type = tibss
server_names = tcp:rtHost2A:5555, ssl:rtHost2B:5571
username = emsServer6
password = myPasswd
project = mfg_process_control
override_lb_mode = enable
delivery_mode = gmd_some

```

Subscribe Mode

Both EMS and SmartSockets allow wildcard subscriptions to collide (for example, in EMS syntax, `foo.*` collides with `foo.bar`; and `foo.*` collides with `*.bar`).

The transport parameter `subscribe_mode` governs SmartSockets message filtering when EMS wildcard subscriptions collide. This section describes the mechanisms of subscription, and the results when import subscriptions collide.

- exact** `exact` instructs `tibemsd` to pass subscriptions to the RTserver *exactly* as the EMS subscribers specify. As a result, subject filtering occurs on the RTserver. Consequently, the RTserver delivers each SmartSockets message with subject `/foo/bar` to this client (`tibemsd`) twice—once for the subscription to `foo.*`, and once for the subscription to `foo.bar`. However, `tibemsd` does not recognize these duplicates as redundant, and delivers two copies to each subscriber. It is illegal to configure `exact` when EMS subscriptions collide.
- all** `all` instructs `tibemsd` to pass the subscription `/...` to the RTserver. As a result, the RTserver delivers *all* messages to this client (only once)—letting `tibemsd` filter the messages. `tibemsd` delivers messages to each subscriber as appropriate, so EMS subscribers do not receive duplicate messages. Because `tibemsd` requests all messages from SmartSockets, an `all` connection carries more data than an `exact` connection.

Destination Name—Syntax and Semantics

- Slash & Dot Separators** This aspect of the mapping between EMS destination names and SmartSockets subjects is straightforward, one-to-one, and bidirectional.

EMS destination names consist of tokens separated by the dot (.) character. SmartSockets subjects consists of tokens preceded by the slash (/) character (like UNIX directory pathnames).

For example, the EMS name `foo.bar.baz` corresponds to the SmartSockets name `/foo/bar/baz`. (Remember that SmartSockets names must begin with a leading slash, but EMS names need not begin with a leading dot. A leading dot indicates an empty element preceding it.)

The slash and dot characters have complementary roles in EMS and SmartSockets. In EMS slash is an ordinary character, while dot is a separator. In SmartSockets slash is a separator, while dot is an ordinary character. To translate names between EMS and SmartSockets, substitute these characters one for another. For example, the EMS name `foo/bar.baz` corresponds to the SmartSockets name `/foo.bar/baz`. However, to avoid confusion, we discourage using either slash or dot as ordinary characters.

Wildcard Star Although both EMS and SmartSockets both interpret the star (*) character as a wildcard, they differ in its semantics. In this aspect, the mapping is not one-to-one.

In EMS, star can match any whole token of a name, but not part of a token. In SmartSockets, star can match part of a token—for example, `/foo/b*/baz` matches `/foo/bar/baz` and `/foo/box/baz`.

If you are familiar with SmartSockets wildcards but not EMS wildcards, see Wildcards on page 44.

Trailing Wildcard In EMS the greater-than (>) character is a wildcard that matches any number of trailing tokens. In SmartSockets a string of three dots (. . .) signifies identical semantics.

Topics

Topics can both export and import messages. Accordingly, you can configure topic definitions (in the configuration file `topics.conf`) with `import` and `export` properties that specify one or more external transports:

- `import`
 - `import` instructs `tibemsd` to import messages that arrive on those transports from SmartSockets, and deliver them to the EMS destination.
- `export`
 - `export` instructs `tibemsd` to take messages that arrive on the EMS destination, and export them to SmartSockets via those transports.



The EMS server *never* re-exports an imported message on the same topic.

(For general information about `topics.conf` syntax and semantics, see `topics` on page 150. You can also configure topics using the administration tool command `addprop topic`.)

Example

For example, the following `tibemsd` commands configure the topic `myTopics.news` to import and export messages on three transports.

```
addprop topic myTopics.news import="SS01,SS02"
addprop topic myTopics.news export="SS01,SS02,SS03"
```

SmartSockets messages with subject `/myTopics/news` arrive at `tibemsd` over the transports `SS01` and `SS02`. EMS clients can receive those messages by subscribing to `myTopics.news`.

EMS messages sent to `myTopics.news` are exported to SmartSockets over all three transports—`SS01`, `SS02` and `SS03`. SmartSockets clients of the corresponding RTservers can receive those messages by subscribing to `/myTopics/news`.

Import Only when Subscribers Exist

When a topic specifies `import` on a connected transport, `tibemsd` imports messages only when the topic has registered subscribers.

Wildcards

Wildcards in the `import` and `export` properties obey EMS syntax and semantics (not SmartSockets syntax and semantics); see Destination Name—Syntax and Semantics on page 102.

Queues

Queues can import messages, but cannot export them.

Configuration

You can configure queue definitions (in the configuration file `queues.conf`) with the `import` property that specify one or more external transports.

- `import` instructs `tibemsd` to import messages that arrive on those transports from SmartSockets, and deliver them to the EMS destination.

(For general information about `queues.conf` syntax and semantics, see `queues` on page 150. You can also configure queues using the administration tool command `addprop queue`.)

Example For example, the following `tibemsdadmin` command configures the queue `myTopics.news` to import messages on the transports `SS01` and `SS02`.

```
addprop topic myQueue.in import="SS01,SS02"
```

SmartSockets messages with subject `/myQueue/in` arrive at `tibemsd` over the transports `SS01` and `SS02`. EMS clients can receive those messages by subscribing to `myQueue.in`.

Import—Start and Stop

When a queue specifies `import` on a connected transport, `tibemsd` immediately begins importing messages to the queue, even when no receivers exist for the queue.

For static queues (configured by an administrator) `tibemsd` continues importing until you explicitly delete the queue.

Wildcards

Wildcards in the `import` property obey EMS syntax and semantics (not SmartSockets syntax and semantics); see `Destination Name—Syntax and Semantics` on page 102.

EMS clients cannot subscribe to wildcard queues—however, you can define wildcard queues in the EMS server for the purpose of property inheritance. That is, you can configure a static queue named `foo.*` and set properties on it, so that child queues named `foo.bar` and `foo.baz` will both inherit those properties.

If you define a queue that imports `foo.*`, `tibemsd` begins importing all matching messages from SmartSockets. As messages arrive, `tibemsd` creates dynamic child queues (for example, `foo.bar` and `foo.baz`) and delivers the messages to them. Notices that `tibemsd` delivers messages to these dynamic child queues even when no subscribers exist to drain them.

Import Issues

This section presents issues associated with importing messages to EMS from SmartSockets—whether on a topic or a queue.

Import Destination Names Must be Unique



When a topic and a queue share the same name, *at most one* of them may set the `import` property. For example, if a topic `foo.bar` and a queue `foo.bar` are both defined, only one may specify the `import` property.

JMSReplyTo

When `tibemsd` imports and translates a SmartSockets message, it sets the `JMSReplyTo` field of the EMS message to the value of the SmartSockets `reply_to` header, so that EMS clients can reply to the message.

Usually this value represents a SmartSockets subject. You must explicitly configure `tibemsd` to create a topic with a corresponding name, which exports messages to SmartSockets.

Guaranteed Delivery



For full end-to-end guaranteed delivery from SmartSockets to EMS, all three of these conditions must be true:

- SmartSockets senders must send messages with guaranteed message delivery (GMD).
- The transport definition must set `topic_import_dm` or `queue_import_dm` (as appropriate) to `TIBEMS_PERSISTENT`.
- A durable subscription for the EMS topic or queue must exist.

For export guarantees, see [Guaranteed Delivery](#) on page 109.

Export Issues

This section presents issues associated with exporting messages from EMS to SmartSockets.

JMSReplyTo

Topics	Consider an EMS message in which the field <code>JMSReplyTo</code> contains a topic. When exporting such a message to SmartSockets, you must explicitly configure <code>tibemsd</code> to import replies from SmartSockets to that reply topic.
Temporary Topics	Consider an EMS message in which the field <code>JMSReplyTo</code> contains a temporary topic. When <code>tibemsd</code> exports such a message to SmartSockets, it <i>automatically</i> arranges to import replies to that temporary topic from SmartSockets; you do not need to configure it explicitly.

Wildcard Subscriptions

Star Wildcard	<p>Both EMS and SmartSockets interpret the star character (*) as a wildcard—but with different semantics. EMS accepts star only as a whole element, which matches a whole element. In contrast, SmartSockets accepts star as part of an element, matching a substring within the element.</p> <p>When a SmartSockets client subscribes to <code>foo.bar*</code>, then configure <code>tibemsd</code> to export the superset <code>foo.*</code>; RTserver narrows the set by delivering only messages that match subscribers.</p> <p>For a full discussion of the differences between EMS and SmartSockets wildcards, see Destination Name—Syntax and Semantics on page 102.</p>
---------------	---

Guaranteed Delivery



For full end-to-end guaranteed delivery to SmartSockets from EMS, both of these conditions must be true:

- EMS senders must send persistent messages.
- The transport definition must set `delivery_mode` to `gmd_some` or `gmd_all` (as appropriate).

To preserve undelivered GMD messages in the EMS undelivered queue, see `preserve_gmd` on page 101.

For import guarantees, see [Guaranteed Delivery](#) on page 108.

Message Translation

JMS Header Fields

EMS supports the ten predefined JMS header fields; see Header Fields on page 56.

Two Special Cases

These two header fields are special cases:

- JMS header `JMSDestination` corresponds to SmartSockets `dest`.
- JMS header `JMSReplyTo` corresponds to SmartSockets `reply_to`.

Import

When importing a SmartSockets message to an EMS message, `tibemsd` does not set any JMS header fields, except for the special cases noted above.

Export

When exporting an EMS message to a SmartSockets message, `tibemsd` groups all the JMS header fields (except for the special cases noted above) into a single submessage within the SmartSockets message. The field `JMSHeaders` contains that submessage. Fields of the submessage map the names of JMS header fields to their values.

`tibemsd` ignores any JMS header fields that are null or absent—it omits them from the exported message.

You can instruct `tibemsd` to suppress the entire header submessage in all exported messages by setting the transport property `export_headers = false`.

JMS Property Fields

Import

When importing a SmartSockets message to an EMS message, `tibemsd` sets these JMS properties:

- `JMS_TIBCO_IMPORTED` gets the value `true`, indicating that the message did not originate from an EMS client.
- `JMS_TIBCO_MSG_EXT` gets the value `true`, indicating that the message *might* contain submessage fields or array fields.
- `JMS_TIBCO_SS_SENDER` gets the value of the SmartSockets `sender` header field (in SmartSockets syntax).

In addition, `tibemsd` maps SmartSockets message properties to EMS properties; for details see SmartSockets Message Properties on page 112.

Export When exporting an EMS message to a SmartSockets message, `tibemsd` groups all the JMS property fields into a single submessage within the SmartSockets message. The field `JMSProperties` contains that submessage. Fields of the submessage map the names of JMS property fields to their values.

`tibemsd` ignores any JMS property fields that are not set, or are set to null—it omits them from the exported message.

You can instruct `tibemsd` to suppress the entire properties submessage in the exported message by setting the transport property `export_properties = false`.

SmartSockets Message Properties

In release 4.1.0 (and later), `tibemsd` maps SmartSockets message headers to EMS message properties on import. Table 15 summarizes the mapping. The first column indicates the EMS property, and the second column indicates the SmartSockets method that gets the corresponding header.

Import The transport parameter `import_ss_headers` governs the import behavior. The third column of Table 15 lists the values of that parameter for which `tibemsd` imports the message property in that row. See `import_ss_headers` on page 100.

Export EMS client programs may modify the values of these properties within imported messages for re-export to SmartSockets. (However, exporting a native EMS message does not carry these properties to SmartSockets.)

Export of these properties depends on the value of the transport parameter `export_properties` on page 100.

When exporting an EMS message to SmartSockets, `tibemsd` maps these properties in reverse. In most cases, the mapping is symmetric—export maps them back to the same SmartSockets header. However, three exceptions (`JMS_TIBCO_SS_SENDER`, `JMS_TIBCO_SS_MESSAGE_ID` and `JMS_TIBCO_SS_SEQ_NUM`) are asymmetric—export maps them to subfields of the field `JMSProperties` within the SmartSockets message. The fourth column of Table 15 indicates this asymmetry.

Table 15 SmartSockets Mapping Message Properties (Import & Export) (Sheet 1 of 2)

EMS Property	SmartSockets Method	Import	Export Asymmetr.
JMS_TIBCO_SS_SENDER	TipcMsgGetSender	none type_num all	Asymmetr.

Table 15 SmartSockets Mapping Message Properties (Import & Export) (Sheet 2 of 2)

EMS Property	SmartSockets Method	Import	Export Asymmetr.
JMS_TIBCO_SS_TYPE_NUM	TipcMsgGetType	type_num all	
JMS_TIBCO_SS_DELIVERY_MODE	TipcMsgGetDeliveryMode	all	
JMS_TIBCO_SS_LB_MODE	TipcMsgGetLbMode	all	
JMS_TIBCO_SS_EXPIRATION	TipcMsgGetExpiration	all	
JMS_TIBCO_SS_PRIORITY	TipcMsgGetPriority	all	
JMS_TIBCO_SS_SENDER_TIMESTAMP	TipcMsgGetSenderTimestamp	all	
JMS_TIBCO_SS_CORRELATION_ID	TipcMsgGetCorrelationId	all	
JMS_TIBCO_SS_USER_PROP	TipcMsgGetUserProp	all	
JMS_TIBCO_SS_MESSAGE_ID	TipcMsgGetMessageId	all	Asymmetr.
JMS_TIBCO_SS_SEQ_NUM	TipcMsgGetSeqNum	all	Asymmetr.

Message Body

tibemsd can export messages with any JMS message body type to TIBCO SmartSockets. Conversely, tibemsd can import messages with any message type from TIBCO SmartSockets.

For information about JMS body types, see Message Bodies on page 57.

For information about the structure of messages, see JMS Message Structure on page 56.

- Import
- When importing a SmartSockets message, tibemsd translates it to one of two EMS message body types:
- If the SmartSockets message contains only *unnamed* fields, then it translates into a `JMSStreamMessage`. The stream contains the values of the unnamed fields in the same order as they appear in the SmartSockets message.
 - If the SmartSockets message contains one or more named fields, then it translates into a `JMSMapMessage`. The map message contains the named fields; the order of the fields is indeterminate.

- Export
- When exporting an EMS message, tibemsd translates it to one of six SmartSockets message types (see Table 16) with the following structure:
- The named field `JMSHeaders` is the first field (omitted when the transport parameter `export_headers` is `false`). It contains a submessage; see JMS Header Fields on page 111.
 - The named field `JMSProperties` is the next field (omitted when the transport parameter `export_properties` is `false`). It contains a submessage; see JMS Property Fields on page 111.
 - The data fields follow the JMS headers and properties (when present). For details about field names and types, see the third column of Table 16.

Table 16 SmartSockets: Mapping Message Types (Export)

JMS Message Type	SmartSockets Message Type	Data Fields
JMSBytesMessage	T_MT_JMS_BYTES	One unnamed field of type T_MSG_FT_BINARY
JMSMapMessage	T_MT_JMS_MAP	Named fields; indeterminate order
JMSObjectMessage	T_MT_JMS_OBJECT	One unnamed field of type T_MSG_FT_BINARY
JMSStreamMessage	T_MT_JMS_STREAM	Unnamed fields in order
JMSTextMessage	T_MT_JMS_TEXT	One unnamed field of type T_MSG_FT_STR
All other JMS message types	T_MT_INFO	No data fields

Data Types

Table 17 presents the mapping between EMS datatypes and SmartSockets datatypes. The mapping is bidirectional, except for a few SmartSockets types that have no corresponding EMS type (for these types the mapping is marked as unidirectional in the middle column of Table 17).

Table 17 *SmartSockets: Mapping Data Types (Sheet 1 of 2)*

EMS	Map	SmartSockets
Boolean		T_MSG_FT_BOOL
Byte		T_MSG_FT_CHAR
Character		T_MSG_FT_INT2
Short		T_MSG_FT_INT2
Integer		T_MSG_FT_INT4
Long		T_MSG_FT_INT8
Float		T_MSG_FT_REAL4
Double		T_MSG_FT_REAL8
Double	<—	T_MSG_FT_TIMESTAMP
String		T_MSG_FT_STR
String	<—	T_MSG_FT_XML
String	<—	T_MSG_FT_UTF8
Byte Array		T_MSG_FT_BINARY
Short Array	<—	T_MSG_FT_BOOL_ARRAY
Short Array		T_MSG_FT_INT2_ARRAY
Integer Array		T_MSG_FT_INT4_ARRAY
Long Array		T_MSG_FT_INT8_ARRAY
Float Array		T_MSG_FT_REAL4_ARRAY
Double Array		T_MSG_FT_REAL8_ARRAY

Table 17 SmartSockets: Mapping Data Types (Sheet 2 of 2)

EMS	Map	SmartSockets
Double Array	<—	T_MSG_FT_TIMESTAMP_ARRAY
Stream Message		T_MSG_FT_MSG
Map Message		(See Import on page 113.)

Destination Names

tibemsd automatically translates destination names when importing or exporting a message; see Slash & Dot Separators on page 102.

When importing, it translates names in the SmartSockets subject and reply_to fields. When exporting, it translates names in the EMS JMSDestination and JMSReplyTo fields.

Chapter 7 **Using the Configuration Files**

This chapter describes configuring TIBCO Enterprise Message Service.

Topics

- *Location of Configuration Files, page 118*
- *Mechanics of Configuration, page 119*
- *Using the Main Configuration File, page 120*
- *Using Other Configuration Files, page 148*

Location of Configuration Files

The installation process places configuration files in two directories:

- `bin/` contains a subset of configuration files suitable for quickly testing the installation.
- `samples/config/` contains the more complete set of sample configuration files. For deployment, we recommend copying files from this directory to a production configuration directory, and modifying those copies.

When selecting a production configuration directory, we recommend using a file system with regular backup commensurate with your need for reliability and disaster recovery. It is essential that the EMS server have both read and write privileges in the configuration directory.



The EMS server reads configuration files only once, when the server starts. It ignores subsequent changes to the configuration files.



You can also change the server configuration with administrative requests, using either `tibemsadmin` (a command line tool), the Java or .NET administrative APIs, or TIBCO Administrator™ (a separate TIBCO product).

When the server validates and accepts an administrative request, it writes the change to the appropriate configuration file as well (overwriting any manual changes to that file). This policy keeps configuration files current in case the server restarts (for example, in a fault-tolerant situation, or after a hardware failure).



Re-installing or updating EMS overwrites the files in the `bin/` and `samples/config/` directories. Do not use these directories to configure your deployment.

Mechanics of Configuration

Configuration Files	The EMS server reads configuration files only once, when the server starts. It ignores subsequent changes to the configuration files.
Administrative Requests	<p>You can also change the server configuration with administrative requests, using either <code>tibemsadmin</code> (a command line tool), the Java or .NET administrative APIs, or TIBCO Administrator™ (a separate TIBCO product).</p> <p>When the server validates and accepts an administrative request, it writes the change to the appropriate configuration file as well (overwriting any manual changes to that file). This policy keeps configuration files current in case the server restarts (for example, in a fault-tolerant situation, or after a hardware failure).</p>

Using the Main Configuration File

The main configuration file controls the characteristics of the TIBCO Enterprise Message Service server. This file is usually named `tibemsd.conf`, but you can specify another file name when starting the server. You can find more information about starting the server in the section *Running the Server* on page 244.

An example of the `tibemsd.conf` file is included in the `bin` directory of TIBCO Enterprise Message Service. You can edit this configuration file with a text editor. There are a few configuration items in this file that can be altered using the administration tool, but most configuration parameters must be set by editing the file (that is, the server does not write out changes to those parameters). See Chapter 8, *Using the Administration Tool*, on page 161 for more information about using the administration tool.

Several parameters accept boolean values. In the description of the parameter, one specific set of values is given (for example, `enable` and `disable`), but all parameters that accept booleans can have the following values:

- `enable`, `enabled`, `true`, `yes`, `on`
- `disable`, `disabled`, `false`, `no`, `off`

Table 18 describes the parameters in `tibemsd.conf`. This table is meant to give a brief description of each parameter.

Table 18 Configuration parameters (Sheet 1 of 28)

Parameter Name	Description
Server Information	
server	Name of server. Server names are limited to at most 64 characters.
password	Password used to log in to other routed server

Table 18 Configuration parameters (Sheet 2 of 28)

Parameter Name	Description
Initialization	
startup_abort_list	<p>This comma-separated list of tokens specifies conditions that cause the server to exit during its initialization sequence. When omitted, the default is the empty list—that is, the server ignores these conditions. You may specify any subset of these tokens:</p> <ul style="list-style-type: none"> SSL—If SSL initialization fails, then exit. TRANSPORTS—If any of the transports cannot be created as specified in the configuration files, then exit. CONFIG_FILES—If any configuration file listed in <code>tibemsd.conf</code> does not exist, then exit. CONFIG_ERRORS—If the server detects any errors while reading the config files, then exit. DB_FILES—If the server cannot find its store files, then exit.
Storage Files	
store	<p>The server stores data in files in this directory.</p> <p>Example</p> <pre>store = /usr/tmp</pre>
store_crc	<p>Specifies whether the EMS server validates CRC checksum data when reading the store files.</p>

Table 18 Configuration parameters (Sheet 3 of 28)

Parameter Name	Description
store_minimum store_minimum_sync store_minimum_async	<p>This set of parameters preallocates disk space for EMS store files. Preallocation occurs when the server first creates a store file.</p> <p>You can specify units of KB, MB, or GB.</p> <p>Zero is a special value, which specifies no minimum preallocation. Otherwise, the value you specify must be greater than or equal to 8MB.</p> <p>If store_minimum_sync or store_minimum_async are absent, they default to store_minimum.</p> <p>If store_truncate is enabled, these parameters limit truncation to minimum values.</p> <p>Example</p> <pre>store_minimum_sync = 32MB</pre>
store_truncate	<p>Specifies whether the EMS server occasionally attempts to truncate the storage files, relinquishing unused disk space.</p> <p>When enabled, the storage files may be truncated, but not below the size specified in the store_minimum parameters.</p>
Flow Control	
flow_control	<p>Specifies whether flow control for destinations is enabled or disabled. By default, flow control is disabled.</p> <p>When flow control is enabled, the flowControl property on each destination specifies the target maximum storage for pending messages on the destination.</p> <p>See Flow Control on page 51 for more information about flow control.</p>

Table 18 Configuration parameters (Sheet 4 of 28)

Parameter Name	Description
Connections and Memory	
<code>max_connections</code>	<p>Maximum number of simultaneous client connections.</p> <p>Set to 0 to allow unlimited simultaneous connections.</p>
<code>max_msg_memory</code>	<p>Maximum memory the server can use for messages.</p> <p>This parameter lets you limit the memory that the server uses for messages, so server memory usage cannot grow beyond the system’s memory capacity.</p> <p>When <code>msg_swapping</code> is enabled, and messages overflow this limit, the server begins to swap messages from process memory to disk. Swapping allows the server to free process memory for incoming messages, and to process message volume in excess of this limit.</p> <p>When the server swaps a message to disk, a small record of the swapped message remains in memory. If all messages are swapped out to disk, and their remains still exceed this memory limit, then the server has no room for new incoming messages. The server stops accepting new messages, and send calls in message producers result in an error. (This situation probably indicates either a very low value for this parameter, or a very high message volume.)</p> <p>Specify units as KB, MB or GB. The minimum value is 8MB. Zero is a special value, indicating no limit.</p> <p>Example</p> <pre>max_msg_memory = 512MB</pre>
<code>msg_swapping</code>	<p>This parameter enables and disables the message swapping feature (described above at <code>max_msg_memory</code>).</p> <p>The default value is enabled, unless you explicitly set it to disabled.</p>

Table 18 Configuration parameters (Sheet 5 of 28)

Parameter Name	Description
<code>reserve_memory = size</code>	<p>When non-zero, the daemon allocates a block of memory for use in emergency situations. When the daemon process exhausts storage resources, it disables clients from producing new messages, and frees this block of memory to allow consumers to continue operation (which tends to free memory).</p> <p>Specify <i>size</i> in units of MB or GB. When non-zero, the minimum block is 16MB. When absent, the default is zero.</p>
<code>msg_pool_block_size size</code> <code>msg_pool_size size</code>	<p>To lessen the overhead costs associated with <code>malloc</code> and <code>free</code>, the server pre-allocates pools of storage for messages. These parameters determine the behavior of these pools. Performance varies depending on operating system platform and usage patterns.</p> <p>The <i>size</i> argument determines the approximate number of internal message structs that a block or pool can accommodate (not the number of bytes).</p> <p><code>msg_pool_block_size</code> instructs the server to allocate an <i>expandable</i> pool. Each time the server exhausts the pool, the server increases the pool by this size, as long as additional storage is available. The value may be in the range 32 to 64K.</p> <p><code>msg_pool_size</code> instructs the server to allocate a <i>fixed</i> pool. After the server exhausts this pool, the server calls <code>malloc</code> each time it requires additional storage. The value may be in the range 16K to 1024M.</p> <p>When neither parameter is present, the default is <code>msg_pool_block_size 128</code> (an expandable pool).</p> <p>These two parameters represent two different and mutually exclusive modes for allocating storage pools. You may specify <i>at most one</i> of these two parameters; it is illegal to set both parameters explicitly.</p>

Table 18 Configuration parameters (Sheet 6 of 28)

Parameter Name	Description
Detecting Network Connection Failure	
<p>This feature lets servers and clients detect network connection failures quickly. This feature is new in release 4.0; it is disabled when either entity is from an earlier release.</p> <p>When these parameters are absent, or this feature is disabled, <code>tibemsd</code> closes a connection only upon the operating system notification.</p>	
<code>client_heartbeat interval</code>	<p>In a server-to-client connection, clients send heartbeats at this interval (in seconds).</p> <p>When omitted or zero, the default is 5 seconds.</p>
<code>client_connection_timeout limit</code>	<p>In a server-to-client connection, if the server does not receive a heartbeat for a period exceeding this limit (in seconds), it closes the connection.</p> <p>We recommend setting this value to approximately 3.5 times the heartbeat interval.</p> <p>Zero is a special value, which disables heartbeat detection in the server (although clients still send heartbeats).</p>
<code>server_heartbeat interval</code>	<p>In a server-to-server connection, this server sends heartbeats at this interval (in seconds).</p> <p>The two servers can be connected either by a route, or as a fault-tolerant pair.</p>
<code>server_connection_timeout limit</code>	<p>In a server-to-server connection, if this server does not receive a heartbeat for a period exceeding this limit (in seconds), it closes the connection.</p> <p>We recommend setting this value to approximately 3.5 times the heartbeat interval of the other server. When the other server or the network are heavily loaded, or when client programs send very large messages, we recommend a larger multiple.</p>

Table 18 Configuration parameters (Sheet 7 of 28)

Parameter Name	Description
Listen Ports	
listen	<p>Format is <i>protocol://servername:port</i></p> <p>Example</p> <p><code>listen=tcp://localhost:7222</code></p> <p>You can use multiple entries for <code>listen</code> if you have computers with multiple interfaces</p> <p>If you are enabling SSL, for example:</p> <p><code>listen=ssl://localhost:7222</code></p>
Authorization	
See Chapter 9, Authentication and Permissions, on page 199 for more information about these parameters.	
authorization	<p>Authorization is disabled by default. If you require that the server verify user credentials and permissions on secure destinations, you must enable this parameter.</p> <p>Example</p> <p><code>authorization= enabled</code></p>

Table 18 Configuration parameters (Sheet 8 of 28)

Parameter Name	Description
<code>user_auth</code>	<p>When a user attempts to authenticate to the EMS server, this parameter specifies the source of authentication information. This parameter can have one or more of the following values (separated by comma characters):</p> <ul style="list-style-type: none"> • <code>local</code>—obtain user authentication information from the local EMS server user configuration. • <code>ldap</code>—obtain user authentication information from an LDAP directory server (see the LDAP-specific configuration parameters). <p>Each time a user attempts to authenticate, the server seeks corresponding authentication information from each of the specified locations in the order that this parameter specifies. The EMS server accepts successful authentication using any of the specified sources.</p>
Routing	
See Chapter 14, Working With Routes, on page 303 for more information about routing.	
<code>routing</code>	<p>Route configuration is in the routes configuration file. This parameter enables or disables routing functionality for this server.</p>
Example	
<code>routing = enabled</code>	
Fault Tolerance Parameters	
See Chapter 13, Fault Tolerance, on page 289 for more information about these parameters.	
<code>ft_active</code>	<p>Name of the active server. If this server can connect to the active server, it will act as a backup server. If this server cannot connect to the active server, it will become the active server.</p>
<code>ft_heartbeat</code>	<p>Heartbeat signal for the active server, in seconds. Default is 3.</p>

Table 18 Configuration parameters (Sheet 9 of 28)

Parameter Name	Description
ft_activation	Activation interval (maximum length of time between heartbeat signals) which indicates that active server has failed. Set in seconds: default is 10. This interval should be set to at least twice the heartbeat interval. Example ft_activation = 60
ft_reconnect_timeout	The amount of time (in seconds) that a backup server waits for clients to reconnect (after it assumes the role of primary server in a failover situation). If a client does not reconnect within this time period, the server removes its state is removed from the shared state files. The default value of this parameter is 60.
ft_ssl_identity	The server’s digital certificate in PEM, DER, or PKCS#12 format. You can copy the digital certificate into the specification for this parameter, or you can specify the path to a file that contains the certificate in one of the supported formats. See File Names for Certificates and Keys on page 275 for more information on file types for digital certificates.
ft_ssl_issuer	Certificate chain member for the server. Supply the entire chain, including the CA root certificate. The server reads the certificates in the chain in the order they are presented in this parameter. The certificates must be in PEM, DER, PKCS#7, or PKCS#12 format. See File Names for Certificates and Keys on page 275 for more information on file types for digital certificates.

Table 18 Configuration parameters (Sheet 10 of 28)

Parameter Name	Description
<code>ft_ssl_private_key</code>	<p>The server's private key. If it is included in the digital certificate in <code>ft_ssl_identity</code>, then this parameter is not needed.</p> <p>This parameter supports private keys in the following formats: PEM, DER, PKCS#12.</p> <p>You can specify the actual key in this parameter, or you can specify a path to a file that contains the key.</p> <p>See File Names for Certificates and Keys on page 275 for more information on file types for digital certificates.</p>
<code>ft_ssl_password</code>	<p>Private key or password for private keys.</p> <p>You can set passwords by way of the <code>tibemsadmin</code> tool. When passwords are set with this tool, the password is obfuscated in the configuration file. See Chapter 8, Using the Administration Tool, on page 161 for more information about using <code>tibemsadmin</code> to set passwords.</p>
<code>ft_ssl_trusted</code>	<p>List of trusted certificates. This sets which Certificate Authority certificates should be trusted as issuers of the client certificates.</p> <p>The certificates must be in PEM, DER, or PKCS#7 format. You can either provide the actual certificates, or you can specify a path to a file containing the certificate chain.</p> <p>See File Names for Certificates and Keys on page 275 for more information on file types for digital certificates.</p>
<code>ft_ssl_rand_egd</code>	<p>The path for the installed entropy gathering daemon (EGD), if one is installed. This daemon is used to generate random numbers for the TIBCO Enterprise Message Service server.</p>

Table 18 Configuration parameters (Sheet 11 of 28)

Parameter Name	Description
ft_ssl_verify_host	<p>Specifies whether the fault-tolerant server should verify the other server’s certificate. The values for this parameter are <code>enabled</code> or <code>disabled</code>. By default, this parameter is <code>enabled</code>, signifying the server should verify the other server’s certificate.</p> <p>When this parameter is set to <code>disabled</code>, the server establishes secure communication with the other fault-tolerant server, but does not verify the server’s identity.</p>
ft_ssl_verify_hostname	<p>Specifies whether the fault-tolerant server should verify the name in the CN field of the other server’s certificate. The values for this parameter are <code>enabled</code> and <code>disabled</code>. By default, this parameter is <code>enabled</code>, signifying the fault-tolerant server should verify the name of the connected host or the name specified in the <code>ft_ssl_expected_hostname</code> parameter against the value in the server’s certificate. If the names do not match, the connection is rejected.</p> <p>When this parameter is set to <code>disabled</code>, the fault-tolerant server establishes secure communication with the other server, but does not verify the server’s name.</p>
ft_ssl_expected_hostname	<p>Specifies the name the server is expected to have in the CN field of the fault-tolerant server’s certificate. If this parameter is not set, the expected name is the hostname of the server.</p> <p>This parameter is used when the <code>ft_ssl_verify_hostname</code> parameter is set to <code>enabled</code>.</p>
ft_ssl_ciphers	<p>Specifies the cipher suites used by the server; each suite in the list is separated by a colon (:). This parameter can use the OpenSSL name for cipher suites or the longer, more descriptive names.</p> <p>See Specifying Cipher Suites on page 281 for more information about the cipher suites available in TIBCO Enterprise Message Service and the OpenSSL names and longer names for the cipher suites.</p>

Table 18 Configuration parameters (Sheet 12 of 28)

Parameter Name	Description
Message Tracking Information	
<code>track_message_ids</code>	<p>Tracks messages by message ID. Default is disabled.</p> <p>Enabling this parameter allows you to display messages using the <code>show message <messageID></code> command in the administration tool.</p>
<code>track_correlation_ids</code>	<p>Tracks messages by correlation ID. Disabled by default.</p> <p>Enabling this parameter allows you to display messages using the <code>show messages <correlationID></code> command in the administration tool.</p>
TIBCO Rendezvous	
See also, Chapter 5, Working With TIBCO Rendezvous, on page 73.	
<code>tibrv_transports</code>	<p>Specifies whether TIBCO Rendezvous transports defined in <code>transports.conf</code> are enabled or disabled.</p> <p>Unless you explicitly set this parameter to <code>enabled</code>, the default value is <code>disabled</code>—that is, all transports are disabled and will neither send messages to external systems nor receive message from them.</p>
<code>tibrv_xml_import_as_string</code>	<p>When importing messages from Rendezvous, <code>tibemsd</code> translates XML fields to byte arrays. Releases earlier than 4.0 erroneously translated them to strings. If your client programs process XML as strings, then enable this parameter to revert to the earlier behavior (strings).</p> <p>When absent, the default value is <code>disabled</code> (byte arrays).</p> <p>(When importing from SmartSockets, XML fields translate to strings. This behavior is correct for SmartSockets, even though it differs from the correct behavior for Rendezvous.)</p>

Table 18 Configuration parameters (Sheet 13 of 28)

Parameter Name	Description
TIBCO SmartSockets	
See also, Chapter 6, Working With TIBCO SmartSockets, on page 95.	
tibss_transports	<p>Specifies whether TIBCO SmartSockets transports defined in <code>transports.conf</code> are enabled or disabled.</p> <p>Unless you explicitly set this parameter to <code>enabled</code>, the default value is <code>disabled</code>—that is, all transports are disabled and will neither send messages to external systems nor receive message from them.</p>
tibss_config_dir	<p>Specifies the directory for SmartSockets configuration files and message files:</p> <ul style="list-style-type: none">• <code>tal_ss.cat</code> is a required file of messages. If it is missing, <code>tibemsd</code> outputs a warning message.• <code>tibems_ss.cm</code> is an optional file of SmartSockets RTclient configuration options. <p>When this parameter is absent, <code>tibemsd</code> searches for these files in its current working directory.</p> <p>For more information about these files, see <i>TIBCO SmartSockets User's Guide</i>.</p>
Tracing and Log File Parameters	
See Chapter 10, Monitoring Server Activity, on page 225 for more information about these parameters.	
logfile	Name and location of the log file.

Table 18 Configuration parameters (Sheet 14 of 28)

Parameter Name	Description
<code>log_trace</code>	<p>Sets the trace preference on the file defined by the <code>logfile</code> parameter. If <code>logfile</code> is not set, the values are stored but have no effect.</p> <p>The value of this parameter is a comma-separated list of trace options. For a list of trace options and their meanings, see Table 35, Server tracing options, on page 228.</p> <p>You may specify trace options in three forms:</p> <ul style="list-style-type: none"> • <code>plain</code> A trace option without a prefix character replaces any existing trace options. • <code>+</code> A trace option preceded by <code>+</code> adds the option to the current set of trace options. • <code>-</code> A trace option preceded by <code>-</code> removes the option from the current set of trace options. <p>Examples</p> <p>The following example sets the trace log to only show messages about access control violations.</p> <pre>log_trace=ACL</pre> <p>The next example sets the trace log to show all default trace messages, in addition to SSL messages, but ADMIN messages are not shown.</p> <pre>log_trace=DEFAULT, -ADMIN, +SSL</pre>

Table 18 Configuration parameters (Sheet 15 of 28)

Parameter Name	Description
logfile_max_size	<p>Specifies the recommended maximum log file size before the log file is rotated. Set to 0 to specify no limit. Use KB, MB, or GB for units (if no units are specified, the file size is assumed to be in bytes).</p> <p>The server periodically checks the size of the current log file. If it is greater than the specified size, the file is copied to a backup and then emptied. The server then begins writing to the empty log file until it reaches the specified size again.</p> <p>Backup log files are named sequentially and stored in the same directory as the current log.</p>
console_trace	<p>Sets trace options for output to stderr. The possible values are the same as for log_trace. However, console tracing is independent of log file tracing.</p> <p>If logfile is defined, you can stop console output by specifying:</p> <pre>console_trace=-DEFAULT</pre> <p>Note that important error messages (and some other messages) are always output, overriding the trace settings.</p> <p>Examples</p> <p>This example sends a trace message to the console when a TIBCO Rendezvous advisory message arrives.</p> <pre>console_trace=RVADV</pre>

Table 18 Configuration parameters (Sheet 16 of 28)

Parameter Name	Description
<code>client_trace={enabled disabled}</code> <code>[target=<location>] [<filter>=<value>]</code>	<p>Administrators can trace a connection or group of connections. When this property is enabled, the server instructs each client to generate trace output for opening or closing a connection, message activity, and transaction activity. This type of tracing does not require restarting the client program.</p> <p>Each client sends trace output to <i><location></i>, which may be either <code>stderr</code> (the default) or <code>stdout</code>.</p> <p>You can specify a filter to selectively trace specific connections. The <i><filter></i> can be <code>user</code>, <code>connid</code> or <code>clientid</code>. The <i><value></i> can be a user name or ID (as appropriate to the filter).</p> <p>When the filter and value clause is absent, the default behavior is to trace all connections.</p> <p>Setting this parameter using the administration tool does not change its value in the configuration file <code>tibemsd.conf</code>; that is, the value does not persist across server restarts unless you set it in the configuration file.</p>
<code>trace_client_host =</code> <code>[hostname address both]</code>	<p>Trace statements related to connections can identify the host by its hostname, its IP address, or both.</p> <p>When absent, the default is <code>hostname</code>.</p>
Statistic Gathering Parameters	
See Chapter 10, Monitoring Server Activity, on page 225 for more information about these parameters.	
<code>server_rate_interval</code>	<p>Sets the interval (in seconds) over which overall server statistics are averaged. This parameter can be set to any positive integer greater than zero.</p> <p>Overall server statistics are always gathered, so this parameter cannot be set to zero. By default, this parameter is set to 1.</p> <p>Setting this parameter allows you to average message rates and message size over the specified interval.</p>

Table 18 Configuration parameters (Sheet 17 of 28)

Parameter Name	Description
statistics	<p>Enables or disables statistic gathering for producers, consumers, destinations, and routes. By default this parameter is set to disabled.</p> <p>Disabling statistic gathering resets the total statistics for each object to zero.</p>
rate_interval	<p>Sets the interval (in seconds) over which statistics for routes, destinations, producers, and consumers are averaged. By default, this parameter is set to 3 seconds. Setting this parameter to zero disables the average calculation.</p>
detailed_statistics	<p>Specifies which objects should have detailed statistic tracking. Detailed statistic tracking is only appropriate for routes, producers that specify no destination, or consumers that specify wildcard destinations. When detailed tracking is enabled, statistics for each destination are kept for the object.</p> <p>Setting this parameter to NONE disabled detailed statistic tracking. You can specify any combination of PRODUCERS, CONSUMERS, or ROUTES to enable tracking for each object. If you specify more than one type of detailed tracking, separate each item with a comma.</p> <p>Examples</p> <p><code>detailed_statistics = NONE</code></p> <p>Turns off detailed statistic tracking.</p> <p><code>detailed_statistics = PRODUCERS, ROUTES</code></p> <p>Specifies detailed statistics should be gathered for producers and routes.</p>
statistics_cleanup_interval	<p>Specifies how long (in seconds) the server should keep detailed statistics if the destination has no activity. This is useful for controlling the amount of memory used by detailed statistic tracking. When the specified interval is reached, statistics for destinations with no activity are deleted.</p>

Table 18 Configuration parameters (Sheet 18 of 28)

Parameter Name	Description
<code>max_stat_memory</code>	<p>Specifies the maximum amount of memory to use for detailed statistic gathering. If no units are specified, the amount is in bytes, otherwise you can specify the amount using KB, MB, or GB as the units.</p> <p>Once the maximum memory limit is reached, the server stops collecting detailed statistics. If statistics are deleted and memory becomes available, the server resumes detailed statistic gathering.</p>
SSL Server Parameters	
See Chapter 12, Using the SSL Protocol, on page 265 for more information about these parameters.	
<code>ssl_dh_size</code>	<p>Size of the Diffie-Hellman key. Can be 512, 768, 1024, or 2048 bits. The default value is 1024.</p> <p>This key is not used for cipher suites available for export.</p>
<code>ssl_server_ciphers</code>	<p>Specifies the cipher suites used by the server; each suite in the list is separated by a colon (:). This parameter must follow the OpenSSL cipher string syntax.</p> <p>For example, you can enable two cipher suites with the following setting:</p> <pre>ssl_server_ciphers = RC4-MD5:RC4-SHA</pre> <p>See Specifying Cipher Suites on page 281 for more information about the cipher suites available in TIBCO Enterprise Message Service and the syntax for specifying them in this parameter.</p>

Table 18 Configuration parameters (Sheet 19 of 28)

Parameter Name	Description
<div>ssl_renegotiate_size</div> <div>Obsolete</div> <div>Key renegotiation is deprecated in release 4.3; it is not supported in release 5.0.</div>	<div>The server renegotiates for a new symmetric key when the cumulative size (in bytes) of the data that the server exchanges with a client reaches this threshold.</div> <div>The minimum value for this parameter is 64Kb. You can specify Kb, Mb, or Gb for the units. For example: ssl_renegotiate_size = 10Gb</div> <div>When neither of the two renegotiation parameters are set, the server does not initiate key renegotiation.</div> <div>For more information, see Renegotiating the Session Key on page 274.</div>
<div>ssl_renegotiate_interval</div> <div>Obsolete</div> <div>Key renegotiation is deprecated in release 4.3; it is not supported in release 5.0.</div>	<div>The server renegotiates for a new symmetric key when the time (in seconds) since the last key negotiation reaches this threshold.</div> <div>The minimum value is 15 seconds. For example, you can set this parameter as follows to renegotiate every 24 hours: ssl_renegotiate_interval = 86400</div> <div>When neither of the two renegotiation parameters are set, the server does not initiate key renegotiation.</div> <div>For more information, see Renegotiating the Session Key on page 274.</div>
<div>ssl_require_client_cert</div>	<div>If this parameter is set to yes, the server only accepts SSL connections from clients that have digital certificates. Connections from clients without certificates are denied.</div> <div>If this parameter is set to no, then connections are accepted from clients that do not have a digital certificate.</div> <div>Whether this parameter is set to yes or no, clients that do have digital certificates are always authenticated against the certificates supplied to the ssl_server_trusted parameter.</div>

Table 18 Configuration parameters (Sheet 20 of 28)

Parameter Name	Description
<code>ssl_use_cert_username</code>	<p>If this parameter is set to <code>yes</code>, a client's user name is always extracted from the CN field of the client's digital certificate, if the digital certificate is specified.</p> <p>The CN field is either a username, an email address, or a web address.</p>
<code>ssl_cert_user_specname</code>	<p>This parameter is useful if clients are required to supply a username, but you wish to designate a special username to use when the client's username should be taken from the client's digital certificate.</p> <p>For example, you may wish all clients to specify their username when logging in. This means the <code>ssl_use_cert_username</code> parameter would be set to <code>no</code>. The username is supplied by the user, and not taken from the digital certificate. However, you may wish one username to signify that the client logging in with that name should have the name taken from the certificate. A good example of this username would be <code>anonymous</code>. All clients logging in as <code>anonymous</code> will have their user names taken from their digital certificates.</p> <p>The value specified by this parameter is the username that clients will use to log in when the username should be taken from their digital certificate. A good example of the value of this parameter would be <code>anonymous</code>.</p> <p>Also, the value of this parameter is ignored if the <code>ssl_use_cert_username</code> parameter is specified. When that parameter is specified, all client usernames are taken from their certificates. This parameter has no effect for users that have no certificate.</p>

Table 18 Configuration parameters (Sheet 21 of 28)

Parameter Name	Description
ssl_server_identity	<p>The server’s digital certificate in PEM, DER, or PKCS#12 format. You can copy the digital certificate into the specification for this parameter, or you can specify the path to a file that contains the certificate in one of the supported formats.</p> <p>This parameter must be specified if any SSL ports are listed in the <code>listen</code> parameter, or if the <code>ssl_enabled</code> parameter is set to <code>true</code>.</p> <p>PEM and PKCS#12 formats allow the digital certificate to include the private key. If these formats are used and the private key is part of the digital certificate, then setting <code>ssl_server_key</code> is optional.</p>
ssl_server_key	<p>The server’s private key. If it is included in the digital certificate in <code>ssl_server_identity</code>, then this parameter is not needed.</p> <p>This parameter supports private keys in the following formats: PEM, DER, PKCS#12.</p> <p>You can specify the actual key in this parameter, or you can specify a path to a file that contains the key.</p>
ssl_password	<p>Private key or password for private keys.</p> <p>This password can optionally be specified on the command line when <code>tibemsd</code> is started.</p> <p>If SSL is enabled, and the password is not specified with this parameter or on the command line, <code>tibemsd</code> will ask for the password upon startup.</p> <p>You can set passwords by way of the <code>tibemsadmin</code> tool. When passwords are set with this tool, the password is obfuscated in the configuration file. See Chapter 8, Using the Administration Tool, on page 161 for more information about using <code>tibemsadmin</code> to set passwords.</p>

Table 18 Configuration parameters (Sheet 22 of 28)

Parameter Name	Description
<code>ssl_server_issuer</code>	<p>Certificate chain member for the server. The server reads the certificates in the chain in the order they are presented in this parameter.</p> <p>The same certificate can appear in multiple places in the certificate chain.</p> <p>The certificates must be in PEM, DER, PKCS#7, or PKCS#12 format.</p>
<code>ssl_server_trusted</code>	<p>List of CA root certificates the server trusts as issuers of client certificates.</p> <p>Specify only CA root certificates. Do not include intermediate CA certificates.</p> <p>The certificates must be in PEM, DER, or PKCS#7 format. You can either provide the actual certificates, or you can specify a path to a file containing the certificate chain.</p> <p>Example</p> <pre>ssl_server_trusted = certs\CA1_root.pem ssl_server_trusted = certs\CA2_root.pem</pre>
<code>ssl_rand_egd</code>	<p>The path for the installed entropy gathering daemon (EGD), if one is installed. This daemon is used to generate random numbers for C clients and the TIBCO Enterprise Message Service server. Java clients do not use this parameter.</p>
<code>ssl_rand_file</code>	<p>File containing random data. This file can be used to generate random numbers.</p>
<code>ssl_crl_path</code>	<p>A non-null value for this parameter activates the server's certificate revocation list (CRL) feature.</p> <p>The server reads CRL files from this directory.</p>
<code>ssl_crl_update_interval</code>	<p>The server automatically updates its CRLs at this interval (in hours).</p> <p>When this parameter is absent, the default is 24 hours.</p>

Table 18 Configuration parameters (Sheet 23 of 28)

Parameter Name	Description
ssl_auth_only	<p>When enabled, the server allows clients to request the use of SSL only for authentication (to protect user passwords). For an overview of this feature, see SSL Authentication Only on page 286.</p> <p>When disabled, the server ignores client requests for this feature. When absent, the default value is disabled.</p>
LDAP General Parameters	
See Chapter 9, Authentication and Permissions, on page 199 for more information about these parameters.	
ldap_url	<p>URL of the external directory server. This can take the following forms:</p> <p>LDAP : //<host> : <tcp_port></p> <p>or</p> <p>LDAPS : //<host> : <ssl_port></p> <p>For example:</p> <p>LDAP : //myLdapServer : 1855</p>
ldap_principal	<p>The distinguished name (DN) of the LDAP administrator. This is the user that the TIBCO Enterprise Message Service sever uses to bind to the LDAP server.</p>
ldap_credential	<p>The password associated with the user defined in the ldap_principal property. This value must be specified and cannot be an empty string.</p>
ldap_cache_enabled	<p>Enables caching of LDAP data.</p>
ldap_cache_ttl	<p>Specifies the maximum time (in seconds) that cached LDAP data is retained before it is refreshed.</p>

Table 18 Configuration parameters (Sheet 24 of 28)

Parameter Name	Description
LDAP Secure Connections	
<code>ldap_conn_type</code>	<p>Specifies the type of connection that the server uses to get LDAP information.</p> <ul style="list-style-type: none"> When this parameter is absent, LDAP connections use TCP (non-secure). For backward compatibility, this is the default setting. <code>ldaps</code>—Use SSL on the LDAP connection (secure). <code>startTLS</code>—Use the startTLS extension to the LDAP version 3 protocol (secure).
<code>ldap_tls_cacert_file</code>	<p>You must specify one of these two parameters for secure connections.</p> <p>This file contains the CA certificate that the TIBCO EMS server trusts to sign the LDAP server's certificate.</p>
<code>ldap_tls_cacert_dir</code>	<p>When there are two or more CA certificates in the verify chain, the server scans this directory for CA certificates.</p>
<code>ldap_tls_ciphers</code>	<p>Optional. You can specify the cipher suite to use for encryption on secure LDAP connections.</p> <p>This parameter must follow the OpenSSL cipher string syntax; see Specifying Cipher Suites on page 281.</p> <p>In addition to the actual cipher names, you may specify cipher quality; for example:</p> <ul style="list-style-type: none"> <code>HIGH</code> <code>HIGH:MEDIUM</code>
<code>ldap_tls_rand_file</code>	<p>When the operating system does not include a random data feature, this file is the source of random data for encryption.</p>
<code>ldap_tls_cert_file</code>	<p>When the LDAP server requires client authentication, use the certificate in this file to identify the TIBCO EMS server.</p>

Table 18 Configuration parameters (Sheet 25 of 28)

Parameter Name	Description
ldap_tls_key_file	<p>When the LDAP server requires client authentication, use the private key in this file.</p> <p>When you plan to start the server remotely, we recommend that you do not password-encrypt the key file.</p>
LDAP User Parameters	
See Chapter 9, Authentication and Permissions, on page 199 for more information about these parameters.	
ldap_user_class	Name of the LDAP object class that stores users.
ldap_user_attribute	Name of the attribute on the user object class that holds the name of the user.
ldap_user_base_dn	Base distinguished name (DN) of the LDAP tree that contains the users.
ldap_user_scope	Specifies how deeply under the base DN to search for users. You can specify <code>onelevel</code> and <code>subtree</code> for this parameter. <code>onelevel</code> specifies to search only one level below the DN, <code>subtree</code> specifies to search all sub-trees.
ldap_user_filter	<p>Optional LDAP search filter for finding a given user name. Use <code>%s</code> as the placeholder for the user name in the filter. For example:</p> <p><code>uid=%s</code></p> <p>The full LDAP search grammar is specified in RFC 2254 and RFC 2251.</p> <p>If unspecified, then a default search filter is generated based on the user object class and user name attribute.</p>
ldap_all_users_filter	<p>An optional LDAP search filter for finding all users beneath the user base DN.</p> <p>If not specified, then a default search filter is generated based on the user object class and user name attribute.</p>

Table 18 Configuration parameters (Sheet 26 of 28)

Parameter Name	Description
LDAP Group parameters	
See Chapter 9, Authentication and Permissions, on page 199 for more information about these parameters.	
<code>ldap_group_base_dn</code>	Base distinguished name (DN) of the LDAP tree that contains groups.
<code>ldap_group_scope</code>	Specifies how deeply under the base DN to search for groups. You can specify <code>onelevel</code> and <code>subtree</code> for this parameter. <code>onelevel</code> specifies to search only one level below the DN, <code>subtree</code> specifies to search all sub-trees.
<code>ldap_group_filter</code>	Optional LDAP search filter for finding a group with a given group name. Use <code>%s</code> as the placeholder for the group name in the filter. The full LDAP search grammar is specified in RFC 2254 2251. If unspecified, then a default search filter is generated based on the group object class and group attribute.
<code>ldap_all_groups_filter</code>	Optional LDAP search filter for finding all groups beneath the group base DN. If unspecified, then a default search filter is generated based on the group object class and group attribute.
<code>ldap_static_group_class</code>	Name of the LDAP object class that stores static groups.
<code>ldap_static_group_attribute</code>	Name of the attribute on the static group object class that holds the name of the group.
<code>ldap_static_member_attribute</code>	Attribute of an LDAP static group object that specifies the distinguished names (DNs) of the members of the group.
<code>ldap_dynamic_group_class</code>	Name of the LDAP object class that stores dynamic groups.
<code>ldap_dynamic_group_attribute</code>	Name of the attribute on the dynamic group object class that holds the name of the group.

Table 18 Configuration parameters (Sheet 27 of 28)

Parameter Name	Description
ldap_dynamic_member_url_attribute	Attribute of the dynamic LDAP group object that specifies the URLs of the members of the dynamic group.
TIBCO Rendezvous Parameters—Deprecated	
These parameters are deprecated. Please configure TIBCO Rendezvous import/export using the289 transports.conf file.	
tibrv_bridge	Enables the bridge between TIBCO Enterprise Message Service and TIBCO Rendezvous. This parameter is disabled by default.
tibrv_service	TIBCO Rendezvous service number. By default, the value is 7500.
tibrv_network	TIBCO Rendezvous network number.
tibrv_daemon	service:hostname:portname. By default, the local daemon is used.
Example	
tcp:hostname:7500	
tibrv_topic_import_dm	Sets the Delivery Mode of the topic (PERSISTENT, NON-PERSISTENT, RELIABLE).
tibrv_queue_import_dm	Sets the Delivery Mode of the queue (PERSISTENT, NON-PERSISTENT, RELIABLE).
tibrv_export_headers	Set this to false if you want to disable exporting JMS headers into TIBCO Rendezvous.
tibrv_export_properties	Set this to false if you want to disable exporting JMS properties into TIBCO Rendezvous.
tibrvcn_enable	Enables a RVCN bridge. Disabled by default.
tibrvcn_name	Name for the transport.
tibrvcn_ledger	Name for file-based ledger.

Table 18 Configuration parameters (Sheet 28 of 28)

Parameter Name	Description
tibrvcn_sync_ledger	Set to <code>true</code> or <code>false</code> . If <code>true</code> , operations that update the ledger do not return until changes are written to the storage medium.
tibrvcn_request_old	Set to <code>true</code> or <code>false</code> . Determines whether a persistent correspondent requires delivery of previously-sent messages of the same name.
tibrvcn_default_ttl	Default time-to-live, in seconds.

Using Other Configuration Files

In addition to the main configuration file, there are several other configuration files used for various purposes. They control configuration for the following:

- users
- groups
- topics
- queues
- access lists
- destination bridges
- routes
- connection factories
- transports
- RVCM listeners
- durable subscribers

These configuration files can be edited by hand, but you can also use the administration tool or the administration APIs to modify some of these files. See Chapter 8, Using the Administration Tool, on page 161 for more information about using the administration tool.

The following sections describe the configuration files.

users

This file defines all users. The format of the file is:

`<username> : <password> : "<description>"`

Item	Description
<code><username></code>	The name of the user

Item	Description
<code><password></code>	<p>Leave this item blank when creating a new user. This is assigned by the system when the user chooses a password. For example:</p> <pre>bob::"Bob Smith"</pre> <p>There is one predefined user, the administrator. The administrator password is not entered in this configuration file, and it will not be assigned by the system. It will remain empty (and therefore <i>not</i> secure) until you set it using the administration tool; see , Assign a Password to the Administrator, on page 165.</p>
<code><description></code>	A string describing the user.

Example

```
admin: $1$urmKVgq78:"Administrator"
Bob::"Bob Smith"
Bill::"Bill Jones"
```

groups

This file defines all groups. The format of the file is:

```
<group-name1>: "<description>"
  <user-name1>
  <user-name2>
<group-name2>: "<description>"
  <user-name1>
  <user-name2>
```

Item	Description
<code><group-name></code>	The name of the group.
<code><description></code>	A string describing the group.
<code><user-name></code>	One or more users that belong to the group.

Example

```
administrators: "TIBCO Enterprise Message Service administrators"
  admin
  Bob
```

topics

This file defines all topics. The format of the file is:

```
<topic-name> <property1>, <property2>, ...
```

For example, you might enter:

```
business.inventory global, import="RV01,RV02", export="RV03",
maxbytes=1MB
```

Only topics listed in this file or topics with names that match the topics listed in this file can be created by the applications. For example, if topic `foo.*` is listed in this file, topics `foo.bar` and `foo.baz` can be created by the application.

Properties of the topic are inherited by all static and dynamic topics with matching names. For example, if `test.*` has the property `secure`, then `test.1` and `test.foo` are also secure. For information on properties that can be assigned to topics, see [Destination Properties](#) on page 34.

For further information on the inheritance of topic properties, refer to [Wildcards * and >](#) on page 44 and [Inheritance of Properties](#) on page 46.

queues

This file defines all queues. The format of the file is:

```
<queue-name> <property1>, <property2>, ...
```

For example, you might enter:

```
test failsafe,secure,prefetch=2
```

Only queues listed in this file or topics with names that match the topics listed in this file can be created by the applications. For example, if queue `foo.*` is listed in this file, queues `foo.bar` and `foo.baz` can be created by the application.

Properties of the queue are inherited by all static and dynamic queues with matching names. For example, if `test.*` has the property `secure`, then `test.1` and `test.foo` are also secure. For information on properties that can be assigned to queues, see [Destination Properties](#) on page 34.

For further information on the inheritance of queue properties, refer to [Wildcards * and >](#) on page 44 and [Inheritance of Properties](#) on page 46.



In the sample file, a `>` wildcard at the beginning of the file allows the applications to create valid queues with any name. A `>` at the beginning of the queue (or topic) configuration file means that name-matching is not required for creation of queues (or topics).

acl

This file defines all permissions on topics and queues for all users and groups.

The format of the file is:

```
TOPIC=<topic> USER=<user> PERM=<permissions>
TOPIC=<topic> GROUP=<group> PERM=<permissions>
QUEUE=<queue> USER=<user> PERM=<permissions>
QUEUE=<queue> GROUP=<group> PERM=<permissions>
ADMIN USER=<user> PERM=<permissions>
ADMIN GROUP=<group> PERM=<permissions>
```

Item	Description
TOPIC	Name of the topic to which you wish to add permissions.
QUEUE	Name of the queue to which you wish to add permissions.
ADMIN	Specifies that you wish to add administrator permissions.
USER	Name of the user to whom you wish to add permissions.
GROUP	Name of the group to which you wish to add permissions. The designation all specifies a predefined group that contains all users.
PERM	<p>Permissions to add.</p> <p>The permissions which can be assigned to queues are send, receive and browse. The permissions which can be assigned to topics are publish, subscribe and durable. The designation all specifies all possible permissions. For information about these permissions, refer to When Permissions Are Checked on page 214 and Inheritance of Permissions on page 46.</p> <p>Administration permissions are granted to users to perform administration activities. See Administrator Permissions on page 216 for more information about administration permissions.</p>

Example

```
ADMIN USER=sys-admins PERM=all
TOPIC=foo USER=user2 PERM=publish,subscribe
TOPIC=foo GROUP=group1 PERM=subscribe
```

bridges

This file defines bridges between destinations. See Destination Bridges on page 48 for more information about destination bridges.

The format of the file is:

```
[ <destinationType> : <destination-name> ]
<destinationType>=<destinationToBridgeTo1> selector="<message-selector>"
<destinationType>=<destinationToBridgeTo2> selector="<message-selector>"
...
```

The <destination-name> can be any specific destination or a wildcard pattern to match multiple destinations.

Item	Description
<destinationType>	The type of the destination. That is, topic or queue.
<destinationToBridgeTo>	One or more names of destinations to which to create a bridge.
selector	<p>This property is optional and specifies a message selector to limit the messages received by the bridged destination.</p> <p>For detailed information about message selector syntax, see documentation for the Message class in <i>TIBCO Enterprise Message Service Java API Reference</i>.</p>

routes

This file defines routes between this TIBCO Enterprise Message Service server and other TIBCO Enterprise Message Service servers.

The format of the file is:

```
[ <route-name> ]
url=<url-string>
zone_name=<zone_name>
zone_type=<zone_type>
[ <selector> ]*
```

[<ssl-prop = value>]*

Item	Description
<route-name>	<route-name> is the name of the passive server (at the other end of the route); it also becomes the name of the route.
url	The URL of the server to and from which messages are routed.
zone_name	<p>The route belongs to the routing zone with this name. When absent, the default value is <code>default_mhop_zone</code> (this default yields backward compatibility with configurations from releases earlier than 4.0).</p> <p>You can set this parameter when creating a route, but you cannot subsequently change it.</p> <p>For further information, see these sections:</p> <ul style="list-style-type: none"> • Zone on page 308 • Configuring Routes and Zones on page 312
zone_type	<p>The zone type is either <code>1hop</code> or <code>mhop</code>. When omitted, the default value is <code>mhop</code>.</p> <p>You can set this parameter when creating a route, but you cannot subsequently change it.</p>
<selector>	<p>Topic selectors (for <code>incoming_topic</code> and <code>outgoing_topic</code> parameters) control the flow of topics along the route.</p> <p>For syntax and semantics, see <i>Selectors for Routing Topic Messages</i> on page 319.</p>
<ssl-prop>	<p>SSL properties for this route.</p> <p>For further information on SSL, refer to Chapter 12, <i>Using the SSL Protocol</i>, page 265.</p>

Example

```
[test_route_2]
url = tcp://server2:7222
```

```
ssl_verify_host = disabled
```

factories

This file defines the connection factories for the internal JNDI names.

The file consists of factory definitions with this format:

```
[<factory-name>]
type = topic | queue
url = <url-string>
metric = connections | byte_rate
clientID = <client-id>
[<prop = value>]*
[<ssl-prop = value>]*
```

(Sheet 1 of 3)

Item	Description
type	Type of the ConnectionFactory. The value can be topic, queue, generic, xatopic, xaqueue, xageneric.
url	<div>This string specifies the servers to which this factory creates connections:</div> <ul style="list-style-type: none">• A single URL specifies a unique server. For example: tcp://host1:8222• A pair of URLs separated by a comma specifies a pair of fault-tolerant servers. For example: tcp://host1:8222,tcp://backup1:8222• A set of URLs separated by vertical bars specifies a load balancing among those servers. For example: tcp://a:8222 tcp://b:8222 tcp://c:8222• You can combine load balancing with fault tolerance. For example: tcp://a1:8222,tcp://a2:8222 tcp://b1:8222,tcp://b2:8222 <div>The load balancing operator () takes precedence over the fault-tolerance operator (,). This example defines two servers (a and b), each of which has a fault-tolerant backup. The client program checks the load on the primary a server and the primary b server, and connects to the one that has the smaller load.</div>

For cautionary information, see Load Balancing on page 156.

(Sheet 2 of 3)

Item	Description
<code>metric</code>	<p>The factory uses this metric to balance the load among a group of servers:</p> <ul style="list-style-type: none"> • <code>connections</code>—Connect to the server with the fewest client connections. • <code>byte_rate</code>—Connect to the server with the lowest byte rate. Byte rate is a statistic that includes both inbound and outbound data. <p>When this parameter is absent, the default metric is <code>connections</code>. For cautionary information, see Load Balancing on page 156.</p>
<code>clientID</code>	The factory associates this client ID string with the connections that it creates.

Properties

Connection factory properties override corresponding properties set using API calls.

<code>connect_attempt_count</code>	A client program attempts to connect to its server (or in fault-tolerant configurations, it iterates through its URL list) until it establishes its first connection to an EMS server. This property determines the maximum number of iterations. When absent, the default is 2.
<code>connect_attempt_delay</code>	When attempting a first connection, the client sleeps for this interval (in milliseconds) between attempts to connect to its server (or in fault-tolerant configurations, iterations through its URL list). When absent, the default is 500 milliseconds.
<code>reconnect_attempt_count</code>	After losing its server connection, a client program configured with more than one server URL attempts to reconnect, iterating through its URL list until it re-establishes a connection with an EMS server. This property determines the maximum number of iterations. When absent, the default is 4.
<code>reconnect_attempt_delay</code>	When attempting to reconnect, the client sleeps for this interval (in milliseconds) between iterations through its URL list. When absent, the default is 500 milliseconds.

(Sheet 3 of 3)

Item	Description
<ssl-prop>	SSL properties for connections that this factory creates. For further information on SSL, refer to Chapter 12, Using the SSL Protocol, page 265.



Example

```
[north_america]
type = topic
url = tcp://localhost:7222,tcp://server2:7222
clientID = "Sample Client ID"
ssl_verify_host = disabled
```

Configuration

To configure connection factories in this file, we recommend using the `tibemsaadmin` tool; see `create_factory` on page 169.

Load Balancing



Do not specify load balancing in situations with durable subscribers.

If a client program that creates durable subscriber connects to server A using a load-balanced connection factory, then server A creates and supports the durable subscription. If the client program exits and restarts, and this time connects to server B, then server B creates and supports a new durable subscription—however, pending messages on server A remain there until the client reconnects to server A.

Do not specify load balancing when your application requires strict message ordering.

Load balancing distributes the message load among multiple servers, which inherently violates strict ordering.

transports

- This file defines transports for importing messages from or exporting messages to external message service:
- For TIBCO Rendezvous, see *Configuring Transports for Rendezvous* on page 76.
 - For TIBCO SmartSockets, see *Configuring Transports for SmartSockets* on page 98.

tibrvcn

This file defines the TIBCO Rendezvous certified messaging (RVCM) listeners for use by topics that export messages to a `tibrvcn` transport. The server preregisters with these listeners when the server starts up so that all messages (including the first message published) sent by way of the `tibrvcn` transport are guaranteed. If the server does not preregister with the RVCM listeners before exporting messages, the listeners are created when the first message is published, but the first message is not guaranteed.

The format of this file is

```
<transport> <listenerName> <subjectName>
```

Item	Description
<code><transport></code>	<p>The name of the transport for this RVCM listener.</p> <p>If you are using the deprecated topic properties and configuration settings for communicating with TIBCO Rendezvous, then do not specify the transport name here. For more information about the <i>deprecated</i> method of exporting to RVCM, TIBCO Rendezvous Parameters—Deprecated on page 146, and Deprecated Properties on page 34.</p>
<code><listenerName></code>	The name of the RVCM listener to which topic messages are to be exported.
<code><subjectName></code>	The RVCM subject name that messages are published to. This should be the same name as the topic names that specify the export property.

Example

```
RVCM01 listener1 foo.bar
RVCM01 listener2 foo.bar.bar
```

durables

This file defines static durable subscribers.

The file consists of lines with either of these formats:

```
<topic-name> <durable-name>
```

<topic-name> <durable-name> <properties>

Item	Description
<topic-name>	The topic of the durable subscription.
<durable-name>	The name of the durable subscriber.
<properties>	A list of properties, separate by commas. Property names and values are described below.

Durable Subscriber Properties	
route	When present, the subscriber is another server, and the <durable-name> is the name of that server. When this property is present, no other properties are permitted.
clientid=<id>	The client ID of the subscriber’s connection.
noLocal	When present, the subscriber does not receive messages published from its own connection.
selector="<sel>"	When present, this selector narrows the set of messages that the durable subscriber receives.

Examples

```
topic1 dName1
topic2 dName2 clientid=myId,nonlocal
topic3 dName3 selector="urgency in ('high,'medium')"
```

topic4 Paris route

- Conflicting Specifications
- When the server detects an conflict between durable subscribers, it maintains the earliest specification, and outputs a warning. Consider these examples:
- A static specification in this file takes precedence over a new durable dynamically created by a client.
 - An existing durable dynamically created by a client takes precedence over a new static durable defined by an administrator.
 - A static durable subscription takes precedence over a client attempting to dynamically unsubscribe (from the same topic and durable name).

Conflict can also arise because of wildcards. For example, if a client dynamically creates a durable subscriber for topic `foo.*`, and an administrator later attempts to define a static durable for topic `foo.1`, then the server detects this conflict and warns the administrator.

Configuration To configure durable subscriptions in this file, we recommend using the `tibemsadmin` tool; see `create durable` on page 169.

If the `create durable` command detects an existing dynamic durable subscription with the same topic and name, it promotes it to a static subscription, and writes a specification to the file `durables.conf`.

Chapter 8

Using the Administration Tool

This chapter gives an overview of commands and use in the administration tool for TIBCO Enterprise Message Service.

Topics

- *Starting the Administration Tool, page 162*
- *Naming Conventions, page 166*
- *Command Listing, page 167*

Starting the Administration Tool

The administration tool is located in your *<installation_path>* /bin directory and is a stand-alone executable named `tibemsadmin` on UNIX and `tibemsadmin.exe` on Windows platforms.

Type **`tibemsadmin -help`** to display information about `tibemsadmin` startup parameters. All `tibemsadmin` parameters are optional.

Table 19 lists options for `tibemsadmin`.

Table 19 *tibemsadmin Options (Sheet 1 of 2)*

Option	Description
<code>-help</code> or <code>-h</code>	Print the help screen.
<code>-script <script-file></code>	<p>Execute the specified text file containing <code>tibemsadmin</code> commands then quit. Any valid <code>tibemsadmin</code> command described in this chapter can be executed.</p> <p>Line breaks within the file delimit each command. That is, every command must be contained on a single line (no line breaks within the command), and each command is separated by a line break.</p>
<code>-server <server-url></code>	Connect to specified server.
<code>-user <user-name></code>	Use this user name to connect to server.
<code>-password <password></code>	Use this password to connect to server.
<code>-ignore</code>	Ignore errors when executing script file. This parameter only ignores errors in command execution but not syntax errors in the script.
<code>-mangle [password]</code>	Mangle the password and quit. Mangled string in the output can be set as a value of server password or server SSL password in the server configuration file. If the password is not entered it is prompted for.
<code>-ssl_trusted <filename></code>	File containing trusted certificate(s). This parameter may be entered more than once if required.
<code>-ssl_identity <filename></code>	File containing client certificate and (optionally) extra issuer certificate(s), and the private key.
<code>-ssl_issuer <filename></code>	File containing extra issuer certificate(s) for client-side identity.

Table 19 *tibemsadmin Options (Sheet 2 of 2)*

Option	Description
-ssl_password <password>	Private key or PKCS#12 password. If the password is required, but has not been specified, it will be prompted for.
-ssl_noverifyhostname	Do not verify hostname against the name on the certificate.
-ssl_hostname <name>	Name expected in the certificate sent by the host.
-ssl_trace	Show loaded certificates and certificates sent by the host.
-ssl_debug_trace	Show additional tracing, which is useful for debugging.



When a command specifies `-user` and `-password`, that information is not stored for later use. It is only used to connect to the server specified in the same command line. The user name and password entered on one command line are not reused with subsequent connect commands entered in the script file or interactively.

Examples

```
tibemsadmin -server "tcp://myhost:7222"
tibemsadmin -server "tcp://myhost:7222" -user admin -password
secret
```

Some options are needed when you choose to make a SSL connection. For more information on SSL connections, refer to Chapter 12, Using the SSL Protocol, page 265.

When You First Start tibemsadmin

The administration tool has a default user with the name `admin`. This is the default user for logging in to the administration tool.

To protect access to the server configuration, you must assign a password to the user `admin`.

Assign a Password to the Administrator

1. Log in and connect to the administration tool, as described directly above.
2. Change the password by entering:

```
set password admin <password>
```

When you restart the administration tool and type `connect`, the administration tool now requires your password before connecting to the server.

For further information about setting and resetting passwords, refer to `set password` on page 177.

Naming Conventions

These rules apply when naming users, groups, topics or queues:

- \$ is illegal at the beginning of the queue or topic names—but legal at the beginning of user and group names.
- Space characters are permitted in a description field—if the entire description field is enclosed in double quotes (for example, "description field").
- Both * and > are wildcards, and cannot be used in names except as wildcards.

For more information about wildcards, see Wildcards on page 44.

- Dot separates elements within a destination name (**foo.bar.***) and can be used only for that purpose.

Command Listing

The command line interface of the administration tool allows you to perform a variety of functions.



Many of the commands listed below accept arguments that specify the names of users, groups, topics or queues. For information about the syntax and naming conventions that apply to these names, see Naming Conventions on page 166.



Note that SSL commands are not listed in this table. SSL commands are listed in several tables in Chapter 12, Using the SSL Protocol, on page 265.

The following is an alphabetical listing of the commands including command syntax and a description of each command.

add member `add member <group_name> <user_name> [,<user2>,<user3>,...]`

Add one or more users to the group. User names that are not already defined are added to the group as external users; see Administration Commands and External Users and Groups on page 207.

addprop factory `addprop factory <factory-name> <properties ...>`

Adds properties to the factory. Property names are separated by spaces.

Factory properties are `url= <url-string>`, `clientID = <client-id>` and SSL parameters.

An example is:

```
addprop factory MyTopicFactory ssl_trusted=cert1.pem
ssl_trusted=cert2.pem ssl_verify_host=disabled
```

For descriptions of factory parameters, see factories on page 154.

For SSL parameters, see Table 41 on page 279.

addprop queue `addprop queue <queue-name> <properties,...>`

Adds properties to the queue. Property names are separated by commas.

addprop route `addprop route <route-name> prop=value[prop-value...]`

Route properties are `url=<url-string>` and SSL parameters.

Note that destination (topic and queue) properties must be separated by commas but properties of routes and factories are separated with spaces.

You can set the `zone_name` and `zone_type` parameters when creating a route, but

you cannot subsequently change them.

For route parameters, see *Configuring Routes and Zones* on page 312.

For the configuration file `routes.conf`, see *routes* on page 152.

addprop topic `addprop topic <topic_name> <properties,...>`

Adds properties to the topic. Property names are separated by commas.

autocommit `autocommit [on|off]`

When `autocommit` is set to **on**, each command causes the change the command made to the configuration files to be saved automatically. When `autocommit` is set to **off**, you must manually use the `commit` command to save configuration changes to the disk.

By default, `autocommit` is set to **on** when interactively issuing commands. If you are running a script, the entire script must complete without errors (or the `ignore` parameter can be specified to ignore errors) for the commit to occur. If there are errors in the script, and the `ignore` parameter is not specified, the administration tool immediately stops the script after the first error and does NOT perform the implicit `commit`.

Entering **autocommit** without parameters displays the current setting of `autocommit` (on or off).

commit `commit`

Commits all configuration changes into files on disk.

compact `compact <store_type> [<max_time>]`

Compacts the database store files.

- To compact the asynchronous file, specify either `a`, `async`, or `asynchronous`.
- To compact the synchronous file, specify either `s`, `sync`, or `synchronous`.

Since compaction can be a lengthy operation, and it blocks other database operations, you may specify a time limit (in seconds). Zero is a special value, which specifies no time limit. When the time limit is absent, the default is zero, and the administration tool asks for confirmation.

We recommend compacting the database store files only when the database Used Space usage is 30% or less (see `show db` on page 187).

connect `connect [<server-url> <admin user name> <password>]`

Connects the administration tool to the server. Any administrator can connect. An administrator is either the admin user, any user in the \$admin group, or any user that has administrator permissions enabled. See Administrator Permissions on page 216 for more information about administrator permissions.

`<server-url>` is usually in the form
`<protocol>://<host-name>:<port-number>`

for example:

```
tcp://myhost:7222
```

The protocol can be `tcp` or `ssl`.

If a user name or password are not provided, the user is prompted to enter a user name and password, or only the password, if the user name was already specified in the command.

You can enter `connect` with no other options and the administrative tool tries to connect to the local server on the default port, which is 7222.

create bridge `create bridge source=<type>:<dest_name> target=<type>:<dest_name>`
 `[selector=<selector>]`

`<type>` is either `topic` or `queue`.

For further information, see `bridges` on page 152.

delete bridge `delete bridge source=<type>:<dest_name> target=<type>:<dest_name>`
 `<type>` is either `topic` or `queue`.

create durable `create durable <topic-name> <durable-name> [<property>, ... ,<property>]`

For descriptions of parameters and properties, and information about conflict situations, see `durables` on page 157.

create factory `create factory factory-name type [URL=url] [clientID=client_id]`
 `[metric=metric] ssl-properties`

Creates a new connection factory.

For descriptions of factory parameters, see `factories` on page 154.

For SSL parameters, see Table 41 on page 279.

create group `create group <group-name> [<group-description>]`

Creates a new group. See `Naming Conventions` on page 166.

create jndiname `create jndiname <new-jndiname> <topic|queue|jndiname> <name>`

Creates a JNDI name for a topic or queue, or creates an alternate JNDI name for a topic that already has a JNDI name.

For example:

```
create FOO jndiname BAR
```

will create new JNDI name FOO referring the same object referred by JNDI name BAR

create queue `create queue <queue-name> [<properties>]`

Creates a queue with the specified name and properties. See Naming Conventions on page 166

Optional queue properties are:

- failsafe
- secure
- global
- maxRedelivery
- exclusive
- import
- flowControl
- trace[=body]
- tibrv_import (deprecated)
- tibrvcn_import (deprecated)
- maxbytes=<number>
- prefetch=<number>
- expiration=<time>
- sender_name
- sender_name_enforced

See Destination Properties on page 34.

create route `create route <name> url=<URL> [<properties ...>]`

Creates a route.

The *name* becomes the name of the new route.

The local server connects to the destination server at the specified URL. If you have configured fault-tolerant servers, you may specify the URL as a comma-separated list of URLs.

You can specify *properties* as a space-separated list of parameter name and value pairs.

You can set the *zone_name* and *zone_type* parameters when creating a route, but you cannot subsequently change them.

If a passive route with the specified *name* already exists, this command promotes it to an *active-active* route; see Active and Passive Routes on page 311.

For route parameters, see Configuring Routes and Zones on page 312.

For the configuration file `routes.conf`, see `routes` on page 152.

create rvcmlistener

```
create rvcmlistener [<transportName>] <name> <subject>
```

Registers an RVCN listener with the server so that any messages forwarded through a `tibrvcn` transport (including the first message sent) are guaranteed for the specified listener. This causes the server to perform the TIBCO Rendezvous call `tibrvcnTransport_AddListener`.

You can optionally specify a transport name to which this RVCN listener applies. If no transport name is specified, the listener is assumed to be for the default RVCN transport.

For more information, see Rendezvous Certified Messaging (RVCN) Parameters on page 76.

create topic

```
create topic <topic-name> [<properties>]
```

Creates a topic with specified name and properties. See Naming Conventions on page 166

Optional topic properties are:

- `failsafe`
- `secure`
- `global`
- `import`
- `export`
- `flowControl`
- `trace[=body]`
- `tibrv_import` (deprecated)
- `tibrvcn_import` (deprecated)
- `tibrv_export` (deprecated)
- `tibrvcn_export` (deprecated)

- `maxbytes=<number>`
- `expiration=<time>`
- `sender_name`
- `sender_name_enforced`

See Destination Properties on page 34.

create user `create user <user-name> [<user-description>][password=<password>]`

Creates a new user. The password is optional, and can be left empty in this command. If the password is not specified in this command, it can be added later using the `set password` command.

See Naming Conventions on page 166.

delete all `delete all <users|groups|topics|queues|durables>
[<topic-name-pattern> | <queue-name-pattern>]`

If used as `delete all <users|groups|topics|queues|durables>` without the optional parameters, the command deletes all users, groups, topics, or queues (as chosen).

If used with a topic or queue, and the optional parameters, such as:

`delete all <topics|queues> <topic-name-pattern> | <queue-name-pattern>`

the command deletes all topics and queues that match the topic or queue name pattern.

delete bridge `delete bridge source=<type>:<dest_name> target=<type>:<dest_name>`
`<type>` is either topic or queue.

delete connection `delete connection <connection-id>`

Deletes the named connection for the client. The connection ID is shown in the first column of the connection description printed by `show connection`.

delete durable `delete durable <durable-name>`

Deletes a durable subscriber.

See also, Conflicting Specifications on page 158.

delete factory `delete factory <factory-name>`

Deletes a factory.

delete group `delete group <group-name>`

Deletes a group.

delete jndiname	<pre>delete jndiname <jndiname></pre> <p>Deletes a jndiname. Notice that deleting the last JNDIname of a connection factory object will remove the connection factory object as well.</p>
delete message	<pre>delete message <messageID></pre> <p>Deletes the message with the specified message ID.</p>
delete queue	<pre>delete queue <queue-name></pre> <p>Deletes a queue.</p>
delete route	<pre>delete route <route-name></pre> <p>Deletes a route.</p>
delete rvcmlistener	<pre>delete rvcmlistener [<transport>] <name> <subject></pre> <p>Unregisters an RVCm listener with the server so that any messages being held for the specified listener in the RVCm ledger are released. This causes the server to perform the TIBCO Rendezvous call <code>tibrvcMTransport_RemoveListener</code>.</p> <p>You can optionally specify a transport name from which this RVCm listener should be deleted. If no transport name is specified, the listener is assumed to be for the default RVCm transport.</p> <p>For more information, see Rendezvous Certified Messaging (RVCm) Parameters on page 76.</p>
delete topic	<pre>delete topic <topic-name></pre> <p>Deletes a topic with specified name.</p>
delete user	<pre>delete user <user-name></pre> <p>Deletes a user.</p>
disconnect	<pre>disconnect</pre> <p>Disconnects the administrative tool from the server.</p>
echo	<pre>echo [on off]</pre> <p>Echo controls the reports that are printed into the standard output. When <code>echo</code> is <code>off</code> the administrative tool only prints errors and the output of queries. When <code>echo</code> is <code>on</code>, the administrative tool report also contains a record of successful command execution.</p> <p>Choosing the parameter <code>on</code> or <code>off</code> in this command controls <code>echo</code>. If <code>echo</code> is entered in the command line without a parameter, it displays the current <code>echo</code> setting (<code>on</code> or <code>off</code>). This command is used primarily for scripts.</p>

The default setting for echo is on.

exit `exit (aliases: quit, q, bye, end)`

Exits the administration tool.

The administrator may choose the `exit` command when there are changes in the configuration have which have not been committed to disk. In this case, the system will prompt the administrator to use the `commit` command before exiting.

grant queue `grant queue <queue> <user-name>|<group-name> <permissions>`

Grants specified permissions to specified user or group on specified queue. The name following the topic name is first checked to be a group name, then a user name.

Specified permissions are added to any existing permissions. Multiple permissions are separated by commas. Enter **all** in the **<permissions>** string if you choose to grant all possible user permissions.

Optional user permissions are:

- `receive`
- `send`
- `browse`

Destination-level administrator permissions can also be granted with this command. The following are administrator permissions for queues.

- `view`
- `create`
- `delete`
- `modify`
- `purge`

For more information, see Chapter 9, Authentication and Permissions, on page 199.

grant topic `grant topic <topic> <user-name>|<group-name> <permissions>`

Grants specified permissions to specified user or group on specified topic. The name following the topic name is first checked to be a group name, then a user name.

Specified permissions are added to any existing permissions. Multiple permissions are separated by commas. Enter **all** in the **<permissions>** string if you choose to grant all possible permissions.

Optional topic permissions are:

- `subscribe`
- `publish`
- `durable`

Destination-level administrator permissions can also be granted with this command. The following are administrator permissions for topics.

- `view`
- `create`
- `delete`
- `modify`
- `purge`

For more information, see Chapter 9, Authentication and Permissions, on page 199.

grant admin `grant admin <group-name | user-name> <admin permissions>`

Grants the specified global administrator permissions to the specified user or group. For a complete listing of global administrator permissions, see Chapter 9, Authentication and Permissions, on page 199.

help `help (aliases: h, ?)`

Usage help.

Enter `help commands` for a command summary.

Enter `help <command>` for help on a specific command.

info `info (alias: i)`

Shows server name and information about the connected server.

purge all queues `purge all queues [<pattern>]`

When used without the optional pattern parameter, this command erases all messages in all queues for all receivers.

When used with the pattern parameter, this command erases all messages in all queues that fit the pattern (for example: `foo.*`).

purge all topics `purge all topics [<pattern>]`

When used without the optional pattern parameter, this command erases all messages in all topics for all subscribers.

When used with the pattern parameter, this command erases all messages in all topics that fit the pattern (for example: `foo.*`).

purge durable	<code>purge durable <durable-name></code> Erases all messages in the topic for a specified durable subscriber
purge queue	<code>purge queue <queue-name></code> Erases all messages in the queue.
purge topic	<code>purge topic <topic-name></code> Erases all messages in the topic.
remove member	<code>remove member <group-name> <user-name> [, <user2> , <user3> , ...]</code> Removes one or more users from a group.
removeprop factory	<code>removeprop factory <factory-name> <properties></code> Removes properties from a factory.
removeprop queue	<code>removeprop queue <queue-name> <properties></code> Removes properties from a queue.
removeprop route	<code>removeprop route <route-name> <properties></code> Removes properties from a route. You cannot remove topic selectors. You can set the <code>zone_name</code> and <code>zone_type</code> parameters when creating a route, but you cannot subsequently change them. For route parameters, see Configuring Routes and Zones on page 312. For the configuration file <code>routes.conf</code> , see routes on page 152.
removeprop topic	<code>removeprop topic <topic-name> <properties></code> Removes properties from a topic.
revoke admin	<code>revoke admin <user> <permissions></code> Revokes the specified global administrator permissions from the specified user. See Chapter 9, Authentication and Permissions , on page 199 for more information about administrator permissions.
revoke queue	<code>revoke queue <queue> <user-name> <group-name> [<permissions>]</code>

If used as `revoke queue <queue-name>` without the optional parameters, the command revokes all permissions for the specified queue.

Revokes specified permissions from a user or group in a specified queue. The name following the queue name is first checked to be a group name, then a user name. If you specify an asterisk (*), all permissions on this queue are removed.

User permissions for queues are `receive`, `send`, `browse`, and `all`.

Administrator permissions for queues are `view`, `create`, `delete`, `modify`, and `purge`.

For more information, see Chapter 9, Authentication and Permissions, on page 199.

revoke topic `revoke topic <topic-name> [<user-name>|<group-name> <permissions>]`

If used as `revoke topic <topic-name>` without the optional parameters, the command revokes all permissions for the specified topic.

When used with the optional parameters, the command revokes specified permissions from a user or group on specified topic. The name following the topic name is first checked to be a group name, then a user name. If you specify an asterisk (*), all permissions on this queue are removed.

Permissions for topics are `subscribe`, `publish`, `durable`, and `all`.

Administrator permissions for topics are `view`, `create`, `delete`, `modify`, and `purge`.

For more information, see Chapter 9, Authentication and Permissions, on page 199.

rotatelog `rotatelog`

Forces the current log file to be backed up and truncated. All entries in the current log file are purged, and the server then starts writing entries to the newly empty log file.

The backup file name is the same as the current log file name with a sequence number appended to the filename. The server queries the current log file directory and determines what the highest sequence number is, then chooses the next highest sequence number for the new backup name. For example, if the log file name is `tibems.log` and there is already a `tibems.log.1` and `tibems.log.2`, the server names the next backup `tibems.log.3`.

set password `set password <user-name> [<password>]`

Sets the password for a user.

If you do not supply a password in the command, the server prompts you to type one.

- To reset a password, type:

```
set password <user-name>
```


Type a new password at the prompt.
- To remove a password, use this command without supplying a password, and press the **Enter** key at the prompt (without typing a password).

After setting passwords (as with other commands) you must use the `commit` command to save the changes to the configuration file.



Passwords are a significant point of vulnerability for any enterprise. We recommend enforcing strong standards for passwords.

For security equivalent to single DES (an industry minimum), security experts recommend passwords that contain 8–14 characters, with at least one upper case character, at least one numeric character, and at least one punctuation character.

set server `set server <parameter=value> [<parameter=value> ...]`

The `set server` command can control many parameters. Multiple parameters are separated by spaces. Table 20 describes the parameters you can set with this command.

Table 20 Set server parameters (Sheet 1 of 6)

Parameter	Description
<code>password[=string]</code>	<p>Sets server password used by the server to connect to other routed servers. If the value is omitted it is prompted for by the administration tool. Entered value will be stored in the main server configuration file in mangled form (but not encrypted).</p> <p>To reset this password, enter the empty string twice at the prompt.</p>
<code>authorization=<enabled disabled></code>	<p>Sets authorization mode.</p> <p>After a transition from disabled to enabled, the server checks ACL permissions for all subsequent requests. While the server requires valid authentication for existing producers and consumers, it does not retroactively reauthenticate them; it denies access to users without valid prior authentication.</p>

Table 20 Set server parameters (Sheet 2 of 6)

Parameter	Description
<code>log_trace=<trace-items></code>	<p>Sets the trace preference on the file defined by the <code>logfile</code> parameter. If <code>logfile</code> is not set, the values are stored but have no effect.</p> <p>The value of this parameter is a comma-separated list of trace options. For a list of trace options and their meanings, see Table 35, Server tracing options, on page 228.</p> <p>You may specify trace options in three forms:</p> <ul style="list-style-type: none"> • plain A trace option without a prefix character replaces any existing trace options. • + A trace option preceded by + adds the option to the current set of trace options. • - A trace option preceded by - removes the option from the current set of trace options. <p>Examples</p> <p>The following example sets the trace log to only show messages about access control violations.</p> <pre>log_trace=ACL</pre> <p>The next example sets the trace log to show all default trace messages, in addition to SSL messages, but ADMIN messages are not shown.</p> <pre>log_trace=DEFAULT, -ADMIN, +SSL</pre>

Table 20 Set server parameters (Sheet 3 of 6)

Parameter	Description
<code>console_trace=<console-trace-items></code>	<p>Sets trace options for output to <code>stderr</code>. The values are the same as for <code>log_trace</code>. However, console tracing is independent of log file tracing.</p> <p>If <code>logfile</code> is defined, you can stop console output by specifying:</p> <pre>console_trace=-DEFAULT</pre> <p>Note that important error messages (and some other messages) are always output, overriding the trace settings.</p> <p>Examples</p> <p>This example sends a trace message to the console when a TIBCO Rendezvous advisory message arrives.</p> <pre>console_trace=RVADV</pre>
<code>client_trace={enabled disabled} [target=<location>] [<filter>=<value>]</code>	<p>Administrators can trace a connection or group of connections. When this property is enabled, the server generates trace output for opening or closing a connection, message activity, and transaction activity. This type of tracing does not require restarting the client program.</p> <p>The server sends trace output to <code><location></code>, which may be either <code>stderr</code> (the default) or <code>stdout</code>.</p> <p>You can specify a filter to selectively trace specific connections. The <code><filter></code> can be <code>user</code>, <code>connid</code> or <code>clientid</code>. The <code><value></code> can be a user name or ID (as appropriate to the filter).</p> <p>When the filter and value clause is absent, the default behavior is to trace all connections.</p> <p>Setting this parameter using the administration tool does not change its value in the configuration file <code>tibemsd.conf</code>.</p>

Table 20 Set server parameters (Sheet 4 of 6)

Parameter	Description
<code>max_msg_memory=<value></code>	<p>Maximum memory the server can use for messages.</p> <p>For a complete description, see <code>max_msg_memory</code> in Table 18 on page 120.</p> <p>Specify units as KB, MB or GB. The minimum value is 8MB. Zero is a special value, indicating no limit.</p> <p>Lowering this value will not immediately free memory occupied by messages.</p>
<code>track_message_ids=<enabled disabled></code>	Enables or disables tracking messages by MessageID.
<code>track_correlation_ids=<enabled disabled></code>	Enables or disables tracking messages by CorrelationID.
<code>ssl_password[=string]</code>	<p>This sets a password for SSL use only.</p> <p>Sets private key or PKCS#12 file password used by the server to decrypt the content of the server identity file. Password stored in mangled form.</p>
<code>ft_ssl_password[=string]</code>	<p>This sets a password for SSL use with Fault Tolerance.</p> <p>Sets private key or PKCS#12 file password used by the server to decrypt the content of the FT identity file. Password stored in mangled form.</p>

Table 20 Set server parameters (Sheet 5 of 6)

Parameter	Description
<code>server_rate_interval=<num></code>	<p>Sets the interval (in seconds) over which overall server statistics are averaged. This parameter can be set to any positive integer greater than zero.</p> <p>Overall server statistics are always gathered, so this parameter cannot be set to zero. By default, this parameter is set to 1.</p> <p>Setting this parameter allows you to average message rates and message size over the specified interval.</p>
<code>statistics=<enabled disabled></code>	<p>Enables or disables statistic gathering for producers, consumers, destinations, and routes. By default this parameter is set to disabled.</p> <p>Disabling statistic gathering resets the total statistics for each object to zero.</p>
<code>rate_interval=<num></code>	<p>Sets the interval (in seconds) over which statistics for routes, destinations, producers, and consumers are averaged. By default, this parameter is set to 3 seconds. Setting this parameter to zero disables the average calculation.</p>
<code>detailed_statistics = < NONE PRODUCERS, CONSUMERS, ROUTES></code>	<p>Specifies which objects should have detailed statistic tracking. Detailed statistic tracking is only appropriate for routes, producers that specify no destination, or consumers that specify wildcard destinations. When detailed tracking is enabled, statistics for each destination are kept for the object.</p> <p>Setting this parameter to NONE disabled detailed statistic tracking. You can specify any combination of PRODUCERS, CONSUMERS, or ROUTES to enable tracking for each object. If you specify more than one type of detailed tracking, separate each item with a comma.</p>

Table 20 Set server parameters (Sheet 6 of 6)

Parameter	Description
<code>statistics_cleanup_interval=<num></code>	Specifies how long (in seconds) the server should keep detailed statistics if the destination has no activity. This is useful for controlling the amount of memory used by detailed statistic tracking. When the specified interval is reached, statistics for destinations with no activity are deleted.
<code>max_stat_memory=<num></code>	<p>Specifies the maximum amount of memory to use for detailed statistic gathering. If no units are specified, the amount is in bytes, otherwise you can specify the amount using KB, MB, or GB as the units.</p> <p>Once the maximum memory limit is reached, the server stops collecting detailed statistics. If statistics are deleted and memory becomes available, the server resumes detailed statistic gathering.</p>
setprop factory	<p><code>setprop factory <factory-name> <properties ...></code></p> <p>Sets the properties for a factory, overriding any existing properties. Multiple properties are separated by spaces.</p>
setprop queue	<p><code>setprop queue <queue-name> <properties, ...></code></p> <p>Sets the properties for a queue, overriding any existing properties. Multiple properties are separated by commas.</p>
setprop route	<p><code>setprop route <route-name> <properties ...></code></p> <p>Sets the properties for a route, overriding any existing properties. Topic and queue names are separated by commas. Multiple properties are separated by spaces.</p> <p>You can set the <code>zone_name</code> and <code>zone_type</code> parameters when creating a route, but you cannot subsequently change them.</p> <p>For route parameters, see <i>Configuring Routes and Zones</i> on page 312.</p> <p>For the configuration file <code>routes.conf</code>, see <i>routes</i> on page 152.</p>
setprop topic	<code>setprop topic <topic-name> <properties></code>

Sets topic properties overriding any existing properties. Multiple properties are separated by commas.

show bridge `show bridge <topic | queue> <name>`

Displays information about the configured bridges for the specified topic or queue. The following is example output for this command:

Target Name	Type	Selector
queue.dest	Q	
topic.dest.1	T	"urgency in ('high', 'medium')"
topic.dest.2	T	

The names of the destinations to which the specified destination has configured bridges are listed in the Target Name column. The type and the message selector (if one is defined) for the bridge are listed in the Type and Selector column.

show bridges `show bridges [type=<topic | queue>] [<pattern>]`

Shows a summary of the destination bridges that are currently configured. The type option specifies the type of destination. For example, `show bridges topic` shows a summary of configured bridges for all topics. The *pattern* specifies a pattern to match for destination names. For example `show bridges foo.*` returns a summary of configured bridges for all destinations that match the name `foo.*`. The type and *pattern* are optional.

The following is example output for this command:

	Source Name	Queue Targets	Topic Targets
Q	queue.source	1	1
T	topic.source	1	2

Destinations that match the specified pattern and/or type are listed in the Source Name column. The number of bridges to queues for each destination is listed in the Queue Targets column. The number of bridges to topics for each destination is listed in the Topic Targets column.

show config `show config`

Shows the configuration parameters for the connected server.

show connections `show connections [type=q|t|s] [host=<hostname>] [user=<username>]
[version] [address]`

Shows connections between clients and server; Table 22 describes the output table. The `type` parameter selects a subset of connections to display; see Table 21. The `hostname` and `username` parameters can further narrow the output to only those connections involving a specific host or user. When the `version` flag is present, the display includes the client's version number.

Table 21 *show connections: type Parameter*

Type	Description
<code>type=q</code>	Show queue connections only.
<code>type=t</code>	Show topic connections only.
<code>type=s</code>	Show system connections only.
<code>absent</code>	Show queue and topic connections, but not system connections.

Table 22 *show connections Table Information (Sheet 1 of 3)*

Heading	Description
<code>L</code>	The type of client. Can be one of the following: <ul style="list-style-type: none"> • <code>J</code> — Java client • <code>C</code> — C client • <code>#</code> — C# client • <code>-</code> — unknown system connection
<code>Version</code>	The TIBCO Enterprise Message Service version of the client.
<code>ID</code>	Unique connection ID. Each connection is assigned a unique, numeric ID that can be used to delete the connection.

Table 22 *show connections Table Information (Sheet 2 of 3)*

Heading	Description
FSXT	<p>Connection type information.</p> <p>The F column displays whether the connection is fault-tolerant.</p> <ul style="list-style-type: none">• - — not a fault-tolerant connection, that is, this connection has no alternative URLs• + — fault-tolerant connection, that is, this connection has alternative URLs <p>The S column displays whether the connection is using the SSL protocol.</p> <ul style="list-style-type: none">• - — connection is not SSL• + — connection is SSL• / — the client uses SSL, but connects by way of an external SSL accelerator to one of the server's TCP ports <p>The X column displays whether the connection is XA.</p> <ul style="list-style-type: none">• - — connection is not XA• + — connection is an XA connection <p>The T column displays the connection type.</p> <ul style="list-style-type: none">• C — generic user connection• T — user TopicConnection• Q — user QueueConnection• A — administrative connection• R — system connection to another route server• F — system connection to the fault-tolerant server
S	Connection started status, + if started, - if stopped.
Host	Connection's host name. (If the name is not available, this column displays the host's IP address.)
Address	<p>Connection's IP address.</p> <p>If you supply the keyword <i>address</i>, then the table includes this column.</p>

Table 22 *show connections Table Information (Sheet 3 of 3)*

Heading	Description
User	Connection user name. If a user name was not provided when the connection was created, it is assigned the default user name anonymous.
ClientID	Client ID of the connection.
Sess	Number of sessions on this connection.
Uptime	Time that the connection has been in effect.

show db `show db <store_type>`

Print a summary of the server's databases.

- To summarize only the asynchronous file, specify either `a`, `async`, or `asynchronous`.
- To summarize only the synchronous file, specify either `s`, `sync`, or `synchronous`.
- To summarize both files, omit the `<store_type>` parameter.

show durable `show durable <durable-name>`

Show information about a durable subscriber.

Table 23 *show durable Table Information (Sheet 1 of 2)*

Heading	Description
Durable Subscriber	Fully qualified name of the durable subscriber. This name concatenates the client ID (if any) and the subscription name (separated by a colon).
Subscription name	Full name of the durable subscriber.
Client ID	Client ID of the subscriber's connection.
Topic	The topic from which the durable subscription receives messages.
Type	<code>dynamic</code> —created by a client <code>static</code> —configured by an administrator

Table 23 *show durable Table Information (Sheet 2 of 2)*

Heading	Description
Status	online offline
Username	Username of the durable subscriber (that is, of the client's connection). If the durable subscriber is currently offline, the value in this column is <code>offline</code> .
Consumer ID	This internal ID number is not otherwise available outside the server.
No Local	<code>enabled</code> —the subscriber does not receive messages sent from its local connection (that is, the same connection as the subscriber). <code>disabled</code> —the subscriber receives messages from all connections.
Selector	The subscriber receives only those messages that match this selector.
Pending Msgs	Number of all messages in the topic. (This count includes the number of delivered messages.)
Delivered Msgs	Number of messages in the topic that have been delivered to the durable subscriber, but not yet acknowledged.
Pending Msgs Size	Total size of all pending messages

show durables

`show durables [<pattern>]`

If a pattern is not entered, this command shows a list of all durable subscribers on all topics.

If a pattern is entered (for example `foo.*`) this command shows a list of durable subscribers on topics that match that pattern.

This command prints a table of information described in Table 24.

Table 24 *show durables* Table Information

Heading	Description
Topic Name	Name of the topic. An asterisk preceding this name indicates a dynamic durable subscriber. Otherwise the subscriber is static (configured by an administrator).
Durable	Full name of the durable subscriber.
User	Name of the user of this durable subscriber. If the durable subscriber is currently offline, the value in this column is <code>offline</code> . For users defined externally, there is an asterisk in front of the user name.
Msgs	Number of pending messages
Size	Total size of pending messages

For more information, see Destination Properties on page 34.

show factory `show factory <factory-name>`

Shows properties of specified factory.

show factories `show factories`

Shows all factories

show jndiname `show jndiname <jndiname>`

Shows the object that the specified name is bound to by the JNDI server.

show jndinames `show jndinames [<type>]`

The optional parameter `<type>` can be:

- destination
- topic
- queue
- factory
- topicConnectionFactory

- queueConnectionFactory

When type is specified only JNDI names bound to objects of the specified type are shown. When type is not specified, all JNDI names are shown.

show group `show group <group-name>`
Shows group name, description, and number of members in the group.
For groups defined externally, there is an asterisk in front of the group name.
Only groups with at least one currently connected user are shown.

show groups `show groups`
Shows all groups.
For groups defined externally, there is an asterisk in front of the group name.
Only groups with at least one currently connected user are shown.

show members `show members <group-name>`
Shows all user members of specified group.

show message `show message <messageID>`
Shows the message for the specified message id.
This command requires that tracking by message ID be turned on using the `track_message_ids` configuration parameter.

show messages `show messages <correlationID>`
Shows the message IDs of all messages with the specified correlation ID set as `JMSCorrelationID` message header field. You can display the message for each ID returned by this command by using the `show message <messageID>` command.
This command requires that tracking by correlation ID be turned on using the `track_correlation_ids` configuration parameter.

show parents `show parents <user-name>`
Shows the user’s parent groups. This command can help you to understand the user’s permissions.

show queue `show queue <queue-name>`

Table 25 *show queue* Table Information (Sheet 1 of 2)

Heading	Description
Queue	Full name of the queue.

Table 25 *show queue Table Information (Sheet 2 of 2)*

Heading	Description
Type	dynamic—created by a client static—configured by an administrator
Properties	A list of property names that are set on the queue, and their values. For an index list of property names, see Destination Properties on page 34.
JNDI Names	A list of explicitly assigned JNDI names that refer to this queue.
Bridges	A list of bridges from this queue to other destinations.
Receivers	Number of consumers on this queue.
Pending Msgs	Number of all messages in the queue. (This count includes the number of delivered messages.)
Delivered Msgs	Number of messages in the queue that have been delivered to a consumer, but not yet acknowledged.
Pending Msgs Size	Total size of all pending messages

show queues `show queues [<pattern-name>]`

If a pattern-name is not entered, this command shows a list of all queues.

If a pattern-name is entered (for example `foo.*`) this command shows a list of queues that match that pattern.

A * appearing before the queue name indicates a dynamic queue.

This command prints a table of information described in Table 26.

Table 26 *show queues Table Information (Sheet 1 of 2)*

Heading	Description
Queue Name	Name of the queue. If the name is prefixed with an asterisk (*), then the queue is temporary or was created dynamically. Properties of dynamic and temporary queues cannot be changed.

Table 26 *show queues Table Information (Sheet 2 of 2)*

Heading	Description
SNFGXIBCT	<p>Prints information on the topic properties in the order (S)ecure (N)sender_name or sender_name_enforced (F)ailsafe (G)lobal e(X)clusive (I)mport (B)ridge (C)flowControl (T)race</p> <p>The characters in the value section show:</p> <ul style="list-style-type: none">- Property not present+ Property is present, and was set on the topic itself* Property is present, and was inherited from another queue <p>Note that inherited properties cannot be removed.</p>
Pre	<p>Prefetch value. if the value is prefixed with an asterisk (*), then it is inherited from another queue or is the default value.</p>
Rcvrs	<p>Number of currently active receivers</p>
Msgs	<p>Number of pending messages</p>
Size	<p>Total size of pending messages</p>

For more information, see Destination Properties on page 34.

show route

`show route <route-name>`
Shows the properties (URL and SSL properties) of a route.

show routes

`show routes`
Shows the properties (URL and SSL properties) of all created routes.
These commands print the information described in Table 27.

Table 27 *show routes Table Information (Sheet 1 of 2)*

Heading	Description
Route	<p>Name of the route.</p>

Table 27 *show routes* Table Information (Sheet 2 of 2)

Heading	Description
T	Type of route: <ul style="list-style-type: none"> • A indicates an active route. • P indicates a passive route.
ConnID	Unique ID number of the connection from this server to the server at the other end of the route. A hyphen (-) in this column indicates that the other server is not running.
URL	URL of the server at the other end of the route.
ZoneName	Name of the zone for the route.
ZoneType	Type of the zone: <ul style="list-style-type: none"> • m indicates a multi-hop zone. • 1 indicates a multi-hop zone.

**show
rvcmlledger**

`show rvcmlledger [<subject or wildcard>]`

This command is provided for backward compatibility with earlier releases and an earlier mechanism for specifying RVCN transports. If you are using the newer mechanism for specifying transports, use the command `show rvcntransportledger` instead.

Displays the TIBCO Rendezvous certified messaging (RVCN) ledger file entries for the specified subject. You can specify a subject name, use wildcards to retrieve all matching subjects, or specify no subject name to retrieve all ledger file entries.

For more information about ledger files and the format of ledger file entries, see TIBCO Rendezvous documentation.

**show
rvcntransportle
dger**

`show rvcntransportledger <transport> [<subject or wildcard>]`

Displays the TIBCO Rendezvous certified messaging (RVCN) ledger file entries for the specified transport and the specified subject. You can specify a subject name, use wildcards to retrieve all matching subjects, or omit the subject name to retrieve all ledger file entries.

For more information about ledger files and the format of ledger file entries, see TIBCO Rendezvous documentation.

show rvcmlisteners	<pre>show rvcmlisteners</pre> <p>Shows all RVCM listeners that have been created using the <code>create rvcmlistener</code> command or by editing the <code>tibrvcml.conf</code> file.</p>
show server	<pre>show server (aliases: info, i)</pre> <p>Shows server name and information about the connected server.</p>
show stat	<pre>show stat consumers [topic=<name> queue=<name>] [user=<name>] [connection=<id>] [total] show stat producers [topic=<name> queue=<name>] [user=<name>] [connection=<id>] [total] show stat route [topic=<name> queue=<name>] [total] [wide] show stat topic <name> [total] [wide] show stat queue <name> [total] [wide]</pre>

Displays statistics for the specified item. You can display statistics for consumers, producers, routes, or destinations. Statistic gathering must be enabled for statistics to be displayed. Also, detailed statistics for each item can be displayed if detailed statistic tracking is enabled. Averages for inbound/outbound messages and message size are available if an interval is specified in the `rate_interval` configuration parameter.

The `total` keyword specifies that only total number of messages and total message size for the item should be displayed. The `wide` keyword displays inbound and outbound message statistics on the same line.

See [Working with Server Statistics](#) on page 238 for a complete description of statistics and how to enable/disable statistic gathering options.

show topic `show topic <topic-name>`

Table 28 *show topic* Table Information (Sheet 1 of 2)

Heading	Description
Topic	Full name of the topic.
Type	dynamic—created by a client static—configured by an administrator
Properties	A list of property names that are set on the topic, and their values. For an index list of property names, see Destination Properties on page 34.
JNDI Names	A list of explicitly assigned JNDI names that refer to this topic.

Table 28 *show topic Table Information (Sheet 2 of 2)*

Heading	Description
Bridges	A list of bridges from this topic to other destinations.
Consumers	Number of consumers on this topic. (This count also includes durable consumers.)
Durable Consumers	Number of durable consumers on this topic.
Pending Msgs	Number of all messages in the topic. (This count includes the number of delivered messages.)
Pending Msgs Size	Total size of all pending messages
The server accumulates the following statistics only when the administrator has enabled statistics. Otherwise these items are zero.	
Total Inbound Msgs	Cumulative count of all messages delivered to the topic.
Total Inbound Bytes	Cumulative total of message size over all messages delivered to the topic.
Total Outbound Msgs	Cumulative count of messages consumed from the topic by consumers. Each consumer of a message increments this count independently of other consumers, so one inbound message results in <i>n</i> outbound messages (one per consumer).
Total Outbound Bytes	Cumulative total of message size over all messages consumed from the topic by consumers. Each consumer of a message contributes this total independently of other consumers.

show topics `show topics [<pattern-name>]`

If a pattern-name is not entered, this command shows a list of all topics.

If a pattern-name is entered (for example `foo.*`) this command shows a list of topics that match that pattern.

This command prints a table of information described in Table 29.

Table 29 *Show topics table information*

Heading	Description
Topic Name	Name of the topic. If the name is prefixed with an asterisk (*), then the topic is temporary or was created dynamically. Properties of dynamic and temporary topics cannot be changed.
SNFGEIBCT	<p>Prints information on the topic properties in the order (S)ecure (N)sender_name or sender_name_enforced (F)ailsafe (G)lobal (E)xport (I)mport (B)ridge (C)flowControl (T)race</p> <p>The characters in the value section show:</p> <ul style="list-style-type: none">- Property not present+ Property is present, and was set on the topic itself* Property is present, and was inherited from another topic <p>Note that inherited properties cannot be removed.</p>
Subs	Number of current subscribers on the topic, including durable subscribers
Durs	Number of durable subscribers on the topic
Msgs	Number of pending messages, including messages to durable receivers
Size	Total size of pending messages

For more information, see Destination Properties on page 34.

show transactions

`show transactions`

Shows all transactions that were created using the XA interface of the C or Java clients. Each transaction is displayed on its own line containing the transaction state followed by the transaction identifier (XID). The transaction state can be one of the following:

- 'A' — active
- 'E' — ended
- 'R' — rollback only

- 'P' —prepared

show transport	<pre>show transport <transport></pre> <p>Displays the configuration for the specified transport defined in <code>transports.conf</code>.</p>
show transports	<pre>show transports</pre> <p>Lists all configured transport names in <code>transports.conf</code>.</p>
show user	<pre>show user <user-name></pre> <p>Shows user name and description. If no user name is specified, this command displays the currently logged in user.</p> <p>For users defined externally, there is an asterisk in front of the user name.</p>
show users	<pre>show users</pre> <p>Shows all users.</p> <p>For users defined externally, there is an asterisk in front of the user name. Only currently connected external users are shown.</p>
showacl group	<pre>showacl group <group-name></pre> <p>Shows all permissions set for a given group. Shows the group and the set of permissions.</p>
showacl queue	<pre>showacl queue <queue-name></pre> <p>Shows all permissions set for a queue. Lists all entries from the <code>acl</code> file. Each entry shows the “grantee” (user or group) and the set of permissions.</p>
showacl topic	<pre>showacl topic <topic-name></pre> <p>Shows all permissions set for a topic. Lists all entries from the <code>acl</code> file. Each entry shows the “grantee” (user or group) and the set of permissions.</p>
showacl user	<pre>showacl user <user-name></pre> <p>Shows all permissions set for a given user. Shows the user and the set of permissions.</p>
shutdown	<pre>shutdown</pre> <p>Shuts down currently connected server.</p>
time	<pre>time [on off]</pre>

Specifying `on` places a timestamp before each command's output. By default, the timestamp is `off`.

transaction commit	<pre>transaction commit <XID></pre> <p>Commits the transaction identified by the transaction ID. The transaction must be in the <code>ended</code> or <code>prepared</code> state. To obtain a transaction ID, issue the <code>show transactions</code> command, and cut and paste the XID into this command.</p>
transaction rollback	<pre>transaction rollback <XID></pre> <p>Rolls back the transaction identified by the transaction ID. The transaction must be in the <code>ended</code>, <code>rollback only</code>, or the <code>prepared</code> state. To obtain a transaction ID, issue the <code>show transactions</code> command, and cut and paste the XID into this command.</p>
updatecrl	<pre>updatecrl</pre> <p>Immediately update the server's certificate revocation list (CRL).</p>
whoami	<pre>whoami</pre> <p>Alias for the <code>show user</code> command to display the currently logged in user.</p>

Chapter 9

Authentication and Permissions

You can create users and assign passwords to the users to control access to the TIBCO Enterprise Message Service server. TIBCO Enterprise Message Service can also be configured to use an external directory (such as an LDAP server) to control access to the server.

You can also assign permissions to users and groups to control actions that can be performed on destinations.

This chapter describes authentication and permissions in TIBCO Enterprise Message Service.

Topics

- *Overview of Users, Groups, and Permissions, page 200*
- *Enabling Access Control, page 203*
- *Users and Groups, page 205*
- *Setting Permissions, page 210*
- *Revoking Permissions, page 213*
- *When Permissions Are Checked, page 214*
- *Administrator Permissions, page 216*

Overview of Users, Groups, and Permissions

TIBCO Enterprise Message Service allows you to control access to the server by creating users and assigning passwords. The server can also authenticate users defined externally (such as an LDAP server).

Permissions apply to the activities a user can perform on each destination (topic and queue). Using permissions you can control which users have permission to send, receive, or browse messages for queues. You can also control who can publish or subscribe to topics, or who can create durable subscriptions to topics. Permissions are stored in the access control list for the server.

Groups allow you to create classes of users and control permissions on a more global level. Rather than granting and revoking permissions on destinations to individual users, you can control destination access at the group level. Users inherit any permissions from each of the groups they belong to, in addition to any permissions that are granted to them directly. Group information can also be retrieved from an external directory, such as an LDAP server.

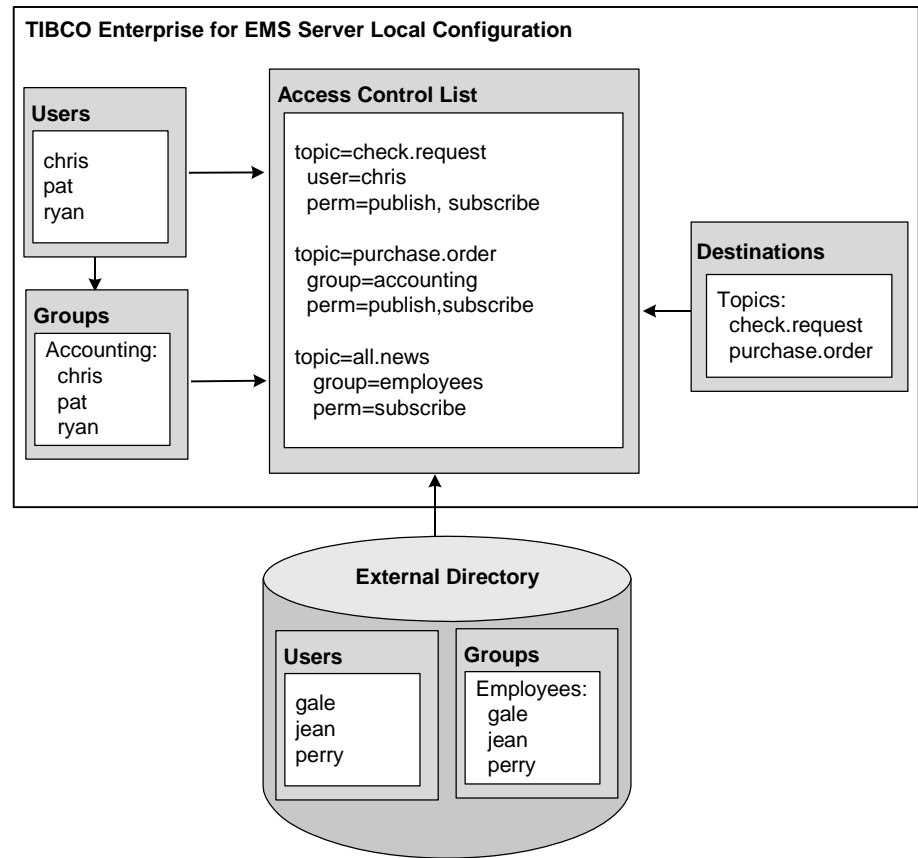


Permissions for all users and groups must be defined in the access control list for the TIBCO Enterprise Message Service server. See [Users and Groups](#) on page 205 for more information about using an external directory service for authenticating users. See [Setting Permissions](#) on page 210 for more information about permissions.

There are also administrator permissions that allow administrators to control which actions users can perform on the server such as create destinations, modify users, and view routes. Administrator permissions can apply globally, or they can be granted on specific destinations.

Figure 15 illustrates the relationships between users, groups and permissions.

Figure 15 Users, groups, and permissions



Externally-configured users and groups are defined and managed using the external directory. Locally-configured users and groups, as well as the access control list, are configured using any of the administration interfaces (editing configuration files, using the administration tool, or the administration APIs).



Access control and Secure Sockets Layer (SSL) have some similar characteristics. SSL allows for servers to require user authentication by way of the user's digital certificate. SSL does not, however, specify any access control at the destination level. SSL and the access control features described in this chapter can be used together or separately to ensure secure access to your system. See Chapter 12, Using the SSL Protocol, on page 265 for more information about SSL.

Setting Up Access Control

The following procedure describes the general process for configuring users, groups, and permissions and where to find more information on performing each step.

1. Enable access control for the system. See [Enabling Access Control](#) on page 203.
2. Optionally enable an external directory for storing users and group information. See [Configuring an External Directory](#) on page 206.
3. Determine the names of the authorized users of the system and create usernames and passwords for these users. See [Users and Groups](#) on page 205.
4. Optionally, set up groups and assign users to groups. See [Users and Groups](#) on page 205.
5. Determine which users need administration permissions, and decide whether administrators can perform actions globally or whether their actions should only be permitted on certain destinations. See [Administrator Permissions](#) on page 216 for more information.
6. Determine which destinations require access control, and enable access control for those destinations. See [Destination Control](#) on page 204.
7. Create the access control list by granting specific permissions to users (or groups) for destinations that need to be secure. See [Setting Permissions](#) on page 210.

Enabling Access Control

Administrators can enable or disable access control for the server. Administrators can also enable and disable permission checking for specific destinations.

Server Control

The property in the main configuration file enables or disables the checking of permissions for all destinations managed by the server. The `authorization` property also enables or disables verification of user names and passwords.



The default setting is disabled. For secure deployments, the administrator must explicitly set authorization to enabled.

When authorization is disabled, the server grants any connection request, and does not check permissions when a client accesses a destination (for example, publishing a message to a topic).

When authorization is enabled, the server grants connections only from valid authenticated users. The server checks permissions for client operations involving secure destinations.

To enable authorization, either edit `tibemspd.conf` (set the `authorization` property to `enabled`, and restart the server). Or you can use the `tibemspdadmin` tool to dynamically enable authorization with the following command:

```
set server authorization=enabled
```

Authorization does affect connections between fault-tolerant server pairs; see [Authorization and Fault-Tolerant Servers](#) on page 299.

Administrators must always log in with the correct administration username and password to perform any administrative function—even when authorization is disabled.

Destination Control

If server authorization is enabled, you can control access to individual destinations by enabling the `secure` property on the destination. When the `secure` property is set on a destination, it instructs the server to check user permissions whenever a user attempts to perform an operation on that destination.



The `secure` property is independent of SSL-level security. The `secure` property controls only basic authentication and permission verification. It does not affect the security of communication between clients and server.

When a destination does not have the `secure` property set, any authenticated user can perform any actions on that topic or queue.

See Destination Properties on page 34 for more information about destination properties.

Users and Groups

The following sections describe users and groups in TIBCO Enterprise Message Service.

Users

Users are specific, named IDs that allow you to identify yourself to the server. When a client logs in, the connect request should be accompanied by a username and the password associated with the username.



In special cases, you may wish to allow anonymous access to the server. In this case, a connect request does not have to supply a username or password. To configure the server to allow anonymous logins, you must create a user named `anonymous` and specify no password. Anonymous logins are not permitted unless the `anonymous` user exists.

Clients logging in anonymously are only able to perform the actions that the `anonymous` user has permission to perform.

There is one predefined user, `admin`. The administrator user is set up when TIBCO Enterprise Message Service is installed, and this user performs administrative tasks, such as creating other users.

You can create and remove users and change passwords by specifying the users in the `users.conf` configuration file, using the `tibemsadmin` tool, or by using the administration APIs. For more information about specifying users in the configuration file, see [users](#) on page 148. For more information about specifying users using the `tibemsadmin` tool, see [Chapter 8, Using the Administration Tool](#), on page 161. For more information on the administration APIs, see the online documentation.

Groups

Groups allow you to create classes of users. Groups make access control administration significantly simpler because you can grant and revoke permissions to large numbers of users with a single operation on the group. Each user can belong to as many groups as necessary. A user's permissions are the union of the permissions of the groups the user belongs to, in addition to any permissions granted to the user directly.

You can create, remove, or add users to groups by specifying the groups in `groups.conf`, using the `tibemsadmin` tool, or by using the administration APIs. For more information about specifying groups in the configuration file, see groups on page 149. For more information about specifying groups using the `tibemsadmin` tool, see Chapter 8, Using the Administration Tool, on page 161. For more information on the administration APIs, see the online documentation.

Configuring an External Directory

You can define user authentication and group information either in EMS server configuration files, or in an external directory (such as an LDAP server).

External User Authentication

TIBCO Enterprise Message Service can be configured to authenticate users stored in an external directory server, such as an LDAP server.

The parameter `user_auth` in `tibemsd.conf` guides the EMS server when authenticating users. When a user attempts to authenticate to the EMS server, this parameter specifies the source of authentication information. This parameter can have one or more of the following values (separated by comma characters):

- `local`—obtain user authentication information from the local EMS server user configuration.
- `ldap`—obtain user authentication information from an LDAP directory server (see the LDAP-specific configuration parameters).

Each time a user attempts to authenticate, the server seeks corresponding authentication information from each of the specified locations in the order that this parameter specifies. The EMS server accepts successful authentication using any of the specified sources.

Group Information

Group information stored in an external directory can also be retrieved by the TIBCO Enterprise Message Service server. Static and dynamic groups are supported and you can configure the TIBCO Enterprise Message Service server to retrieve either or both.

Administration Commands and External Users and Groups

You can perform administrative commands on users and groups defined either locally (in the EMS server's local configuration files) or in an external LDAP. Furthermore, you can combine users and groups that are defined in different locations (for example, you can to grant and revoke permissions for users and groups defined in an LDAP, or add LDAP-defined users to locally-defined groups).



Combining authentication sources requires that the configuration parameter `user_auth` includes both `ldap` and `local`.

When you attempt to view users and groups using the `show user/s` or `show group/s` commands, any users and groups that exist in external directories have an asterisk next to their names. Users and groups from external directories will only appear in the output of these commands in the following situations:

- an externally-defined user successfully authenticates
- a user belonging to an externally-defined group successfully authenticates
- an externally-defined user has been added to a locally-defined group
- permissions on a topic or queue have been granted to an externally-defined user or group

Therefore, not all users and groups defined in the external directory may appear when the `show user/s` or `show group/s` commands are executed. Only the users and groups that meet the above criteria at the time the command is issued will appear.

You can create users and groups with the same names as externally-defined users and groups. If a user or group exists in the server's configuration and is also defined externally, the local definition of the user takes precedence. Locally-defined users and groups will not have an asterisk by their names in the `show user/s` or `show group/s` commands.

You can also issue the `delete user` or `delete group` command to delete users and groups from the local server's configuration. The permissions assigned to the user or group are also deleted when the user or group is deleted. If you delete a user or group that is defined externally, this deletes the user or group from the server's memory and deletes any permissions assigned in the access control list, but it has no effect on the external directory. The externally-defined user can once again log in, and the user is created in the server's memory and any groups to which the user belongs are also created. However, any permissions for the user or group have been deleted and therefore must be re-granted.

Using LDAP Directory Servers

TIBCO Enterprise Message Service has been tested with the following external directory servers:

- Netscape/SunOne iPlanet Directory Server version 5.1
- Microsoft Active Directory shipped as part of the Windows 2000 Server

However, you should be able to use any external directory server that is compliant with LDAP V2.

Table 18, Configuration parameters, on page 120 describes the complete list of configuration parameters for configuring an external directory server. Table 30 describes parameter settings for default configurations of popular LDAP servers.

Table 30 Default configuration for popular LDAP servers (Sheet 1 of 2)

External Directory Server	Parameter Configuration
iPlanet	<pre>ldap_principal = cn=Directory Manager ldap_user_class = Person ldap_user_attribute = uid ldap_user_base_dn = ou=people, o=<your_organization> ldap_user_filter = (&(uid=%s)(objectclass=person)) ldap_group_base_dn = "ou=groups, o=<your_organization> ldap_group_filter = ((&(cn=%s)(objectclass=groupofUniqueNames))(& (cn=%s)(objectclass=groupOfURLs))) ldap_static_group_class = groupofuniquenames ldap_staic_group_attribute = cn ldap_static_member_attribute = uniquemember ldap_dynamic_group_class = groupofURLs ldap_static_group_member_filter = (&(uniquemember=%s)(objectclass=groupofuniquen ames)) ldap_dynamic_group_class = groupofURLs ldap_dynamic_group_attribute = cn ldap_dynamic_member_url_attribute = memberURL</pre>

Table 30 Default configuration for popular LDAP servers (Sheet 2 of 2)

External Directory Server	Parameter Configuration
Active Directory	<pre> ldap_principal = CN=Administrator, CN=Users, DC=<your_domain> ldap_user_class = user ldap_user_attribute = cn ldap_user_filter = (&(cn=%s)(objectclass=user)) ldap_group_filter = (&(cn=%s)(objectclass=group)) ldap_static_group_class = group ldap_static_group_attribute = cn ldap_static_member_attribute = member ldapt_static_group_member_filter = (&(member=%s)(objectclass=group)) </pre>
Open LDAP	<pre> ldap_user_class = person ldap_user_attribute = cn ldap_user_base_dn = ou=people, dc=<your_domain_component>, dc=<your_domain_component> ldap_user_filter = (&(cn=%s)(objectclass=user)) ldap_group_base_dn = ou=groups, dc=<your_domain_component>, dc=<your_domain_component> ldap_group_filter = (&(cn=%s)(objectclass=groupofnames)) ldap_static_group_class = groupofnames ldap_static_group_attribute = cn ldap_static_member_attribute = member ldap_static_group_member_filter = (&(member=%s)(objectclass=groupofnames)) </pre>
Novell	<pre> ldap_user_class = person ldap_user_attribute = cn ldap_user_base_dn = ou=people, o=<your_organization> ldap_user_filter = (&(cn=%s)(objectclass=person)) ldap_group_base_dn = ou=groups, o=<your_organization> ldap_group_filter = (&(cn=%s)(objectclass=groupofnames)) ldap_static_group_class = grouponames ldap_static_group_attribute = cn ldap_static_member_attribute = uniquemember ldap_static_group_member_filter = (&(uniquemember=%s)(objectclass=groupofnames)) </pre>

Setting Permissions

Permissions are stored in the access control list and determine the actions a user can perform on a destination. A user’s permissions are the union of the permissions granted explicitly to that user along with any permissions the user receives by belonging to a group.

When granting permissions, you specify the user or group to whom you wish to grant the permission, the name of the destination, and the permission(s) to grant. Granting permissions is an action that is independent from both the authorization server parameter, and the secure property of the relevant destinations. The currently granted permissions are stored in the access control file, however, the server enforces them only if the authorization is enabled, and only for secure destinations.



When setting permissions for users and groups defined externally, user and group names are case-sensitive. Make sure you use the correct case for the name when setting the permissions.

Permissions can only be granted by users that have the appropriate administrator permissions. See Administrator Permissions on page 216 for more information.

You can specify either explicit destination names or wildcard destination names. See Inheritance of Permissions on page 211 for more information on wildcard destination names and permissions.

Topics and queues each have associated permissions. Table 31 describes the permissions specific to queues. Table 32 describes the permissions specific to topics.

Table 31 Queue Permission

Name	Description
receive	permission to create queue receivers
send	permission to create queue senders
browse	permission to create queue browsers

Table 32 Topic Permission (Sheet 1 of 2)

Name	Description
subscribe	permission to create non-durable subscribers on the topic

Table 32 Topic Permission (Sheet 2 of 2)

Name	Description
<code>publish</code>	permission to publish on the topic
<code>durable</code>	permission to create durable subscribers on the topic
<code>use_durable</code>	permission to use existing durable subscribers on the topic, but <i>not</i> to create nor configure them

You assign permissions either by specifying them in the `acl.conf` file, using the `tibemsadmin` tool, or by using the administration APIs.

Example of Setting Permissions

The user `bob` has the following permission recorded in the `acl.conf` file:

```
USER=bob TOPIC=foo PERM=subscribe,publish
```

This set of permissions means that `bob` can subscribe to topic `foo` and publish messages to it, but `bob` cannot create durable subscribers to `foo`.

If `bob` is a member of group `engineering` and the group has the following entry in the `acl` file:

```
GROUP=engineering TOPIC=bar PERM=subscribe,publish
```

then `bob` can publish and subscribe to topics `foo` and `bar`.

If both the user `bob` and the group `engineering` have entries in the `acl.conf` file, then `bob` has permissions that are a union of all permissions set for `bob` directly and the permissions of the group `engineering`.

Inheritance of Permissions

When you grant permissions to users for topics or queues with wildcard specifications, all created topics and queues that match the specification will have the same granted permissions as the permissions on the parent topic. If there are multiple parent topics, the user receives the union of all parent topic permissions for any child topic. You can add permissions to a user for topics or queues that match a wildcard specification, but you cannot remove permissions.

For example, you can grant user `Bob` the `browse` permission on queue `foo.*`. The user `Bob` receives the `browse` permission on the `foo.bar` queue, and you can also grant `Bob` the `send` permission on the `foo.bar` queue. However, you cannot take away the inherited `browse` permission from `Bob` on the `foo.bar` queue.

See Wildcards on page 44 for more information about wildcards in destination names.

Revoking Permissions

Administrators can revoke permissions for users to create consumers on a destination. Without permission, the user cannot create new consumers for a destination—however, existing consumers of the destination continue to receive messages.

You can only revoke a permission that is granted directly. That is, you cannot revoke a permission from a user that the user receives from a group. Also, you cannot revoke a permission that is inherited from a parent topic. The `revoke` command in `tibemsadmin` can only remove items from specific entries in the `acl.conf` file. The `revoke` command cannot remove items that are inherited from other entries.

You can revoke permissions in several ways:

- Remove or edit entries in the `acl.conf` file.
- Use the `revoke` commands in `tibemsadmin`; see page 176.
- Use the administration APIs.

When Permissions Are Checked

If permissions are enforced (that is, the `authorization` configuration property is set, and the `secure` property is set for the destination), the server checks them when a user attempts to perform an operation on a destination. For example, create a subscription to a topic, send a message to a queue, and so on. Since permissions can be granted or revoked dynamically, the server checks them each time an operation is performed on a destination (and each time a consumer or producer is created).

For specific (non-wildcard) destination names, permissions are checked when a user performs one of the following actions:

- creates a subscription to a topic
- attempts to become a consumer for a queue
- publishes or sends a message to a topic or queue

A user cannot create or send a message to a destination for which he or she has not explicitly been granted the appropriate permission. So, before creating or sending messages to the destination, a user must be granted permissions on the destination.

However, for wildcard topic names (queue consumers cannot specify wildcards), permissions are not checked when users create non-durable subscriptions. Therefore, a user can create a subscription to topic `foo.*` without having explicit permission to create subscriptions to `foo.*` or any child topics. This allows administrators to grant users the desired permissions after the user's application creates the subscriptions. You may wish to allow users to subscribe to unspecific topics, then grant permission to specific topics at a later time. Users are not able to receive messages based on their wildcard subscriptions until permissions for the wildcard topic or one or more child topics are granted.



When creating a durable subscriber, users must have the `durable` permission explicitly set for the topic they are subscribing to. For example, to create a durable subscriber to topic `foo.*`, the user must have been granted the `durable` permission to create durable subscriptions for topic `foo.*`.

Example of Permission Checking

This example walks through a scenario for granting and revoking permissions to a user, and describes what happens as various operations are performed.

1. User bob is working with a TIBCO Enterprise Message Service application that subscribes to topics and displays any messages sent to those topics.

2. User bob creates a subscription to `user.*`. This topic is the parent topic of each user. Messages are periodically sent to each user (for example, messages are sent to the topic `user.bob`). Because the same application is used by many users, the application creates a subscription to the parent topic.
3. User bob creates a subscription to topic `corp.news`. This operation fails because bob has not been granted access to that topic yet.
4. A message is sent to the topic `user.bob`, but the application does not receive the message because bob has not been granted access to the topic yet.
5. The administrator, as part of the daily maintenance for the application, grants access to topics for new users. The administrator grants the subscribe permission to topic `user.bob` and `corp.*` to user bob. These grants occur dynamically, and user bob is now able to receive messages sent to topic `user.bob` and can subscribe to topic `corp.news`.
6. The administrator sends a message on the topic `user.bob` to notify bob that access has been granted to all `corp.*` topics.
7. The application receives the new message on topic `user.bob` and displays the message.
8. User bob attempts to create a subscription for topic `corp.news` and succeeds.
9. A message is sent to topic `corp.news`. User bob's application receives this message and displays it.
10. The administrator notices that bob is a contractor and not an employee, so the administrator revokes the subscribe permission on topic `corp.*` to user bob. The subscription to `corp.news` still exists for user bob's application, but bob cannot create any new subscriptions to children of the `corp.*` topic.

Administrator Permissions

Administrators are a special class of users that can manage the TIBCO Enterprise Message Service server. Administrators create, modify, and delete users, destinations, routes, factories, and other items. In general, administrators must be granted permission to perform administration activities when using the administration tool or API. Administrators can be granted global permissions (for example, permission to create users or to view all queues), and administrators can be granted permissions to perform operations on specific destinations (for example, purging a queue, or viewing properties for a particular topic).



Administrator permissions control what administrators can view and change in the server only when using the administration tool or API. Administrator commands create entries in each of the configuration files (for example, `tibemsd.conf`, `acl.conf`, `routes.conf`, and so on).

You should control access to the configuration files so that only certain system administrators can view or modify the configuration files. If a user can view or modify the configuration files, setting permissions to control which destination that user can manage would not be enforced when the user manually edits the files.

Use the facilities provided by your Operating System to control access to the server's configuration files.

Administrators cannot be defined in an external directory. Administrators must be created using the administration tool, the administration APIs, or in the configuration files.

Predefined Administrative User and Group

There is a special, predefined user named `admin` that can perform any administrative action. You cannot grant or revoke any permissions to `admin`. This user is created when the server is installed, and you must change the password for `admin` immediately after installation. For more information about changing the `admin` password, see *When You First Start tibemsaadmin* on page 164.

There is also a special group named `$admin` for system administrator users. When a user becomes a member of this group, that user receives the same permissions as the `admin` user. You cannot grant or revoke administrator permissions from any user that is a member of the `$admin` group. You should only assign the overall system administrator(s) to the `$admin` group.

Granting and Revoking Administration Permissions

You grant and revoke administrator permissions to users using the `grant` and `revoke` commands in `tibemsadmin`. You can either grant global administrator permissions or permissions on specific destinations. See Global Administrator Permissions on page 218 for a complete list of global administrator permissions. See Destination-Level Permissions on page 221 for a description of administrator permissions for destinations.

Global and destination-level permissions are granted and revoked separately using different administrator commands. See Command Listing on page 167 for the syntax of the `grant` and `revoke` commands.

If a user has both global and destination-level administrator permissions, the actions that user can perform are determined by combining all global and destination-level administrator permissions granted to the user. For example, if an administrator is granted the `view-destination` permission, that administrator can view information about all destinations, even if the `view` permission is not granted to the administrator for specific destinations.

The `admin` user or all users in the `$admin` group can grant or revoke any administrator permission to any user. All other users must be granted the `change-admin-acl` permission and the `view-user` and/or the `view-group` permissions before they can grant or revoke administrator permissions to other users.

If a user has the `change-admin-acl` permission, that user can only grant or revoke permissions that have been granted to the user. For example, if user `BOB` is not part of the `$admin` group and he has only been granted the `change-admin-acl` and `view-user` permissions, `BOB` cannot grant any administrator permissions except the `view-user` or `change-admin-acl` permissions to other users.

Users have all administrator permissions that are granted to any group to which they belong. You can create administrator groups, grant administrator permissions to those groups, and then add users to each administrator group. The users will be able to perform any administrative action that is allowed by the permissions granted to the group to which the user belongs.

Any destination-level permission granted to a user or group for a wildcard destination is inherited for all child destinations that match the parent destination.

If protection permissions are set up, administrators can only grant or revoke permissions to other users that have the same protection permission as the administrator. See Protection Permissions on page 223 for more information about protection permissions.

Enforcement of Administrator Permissions

An administrator can only perform actions for which the administrator has been granted permission. Any action that an administrator performs may be limited by the set of permissions granted to that administrator.

For example, an administrator has been granted the `view` permission on the `foo.*` destination. This administrator has not been granted the global `view-destination` permission. The administrator is only able to view destinations that match the `foo.*` parent destination. If this administrator is granted the global `view-acl` permission, the administrator is only able to view the access control list for destinations that match the `foo.*` parent. Any access control lists for other destinations are not displayed when the administrator performs the `showacl topic` or `showacl queue` commands.

In some cases, enforcement of permissions causes the administrator to see little or no output for any `view` administrator commands. An administrator will receive an error when attempting to view a specific item for which the administrator does not have permission. For example, if the administrator above issues the `showacl queue` command, and there are no queues named `foo.*`, the command executes and returns no output. However, if the administrator issues the `showacl queue bar.foo` command, the administrator receives a “Not authorized to execute command” error because the administrator is not authorized to view any destination except those that match `foo.*`.



An administrator can always change his/her own password, even if the administrator is not granted the `change-user` permission.

An administrator can always view his/her own permissions by issuing the `showacl <username>` command, even if the administrator is not granted the `view-acl` permission.

Global Administrator Permissions

Certain permissions allow administrators to perform global actions, such as creating users or viewing all queues.

Table 33 describes the global administrator permissions.

Table 33 Global administrator permissions (Sheet 1 of 2)

Permission	Allows Administrator To...
all	Perform all administrative commands.
change-acl	Grant and revoke user-level permissions.
change-admin-acl	Grant and revoke administrative permissions.
change-bridge	Create and delete destination bridges.
change-connection	Delete connections.
create-destination	Create any destination.
modify-destination	Modify any destination.
delete-destination	Delete any destination.
change-durable	Delete durable subscribers.
change-factory	Create, delete, and modify factories.
change-group	Create, delete, and modify groups.
change-message	Delete messages stored in the server.
change-route	Create, delete, and modify routes
change-server	Modify server parameters.
change-user	Create, delete, and modify users.
purge-destination	Purge destinations.
purge-durable	Purge durable subscribers.
shutdown	Shutdown the server.
view-acl	View user-level permissions.
view-admin-acl	View administrative permissions.

Table 33 Global administrator permissions (Sheet 2 of 2)

Permission	Allows Administrator To...
view-all	View any item that can be administered (for example, users, groups, topics, and so on).
view-connection	View connections, producers and consumers.
view-bridge	View destination bridges.
view-destination	View destination properties and information.
view-durable	View durable subscribers. To view a durable subscriber, you must also have view-destination permission (because information about a durable subscriber includes information about the destination to which it subscribes.)
view-factory	View factories.
view-group	View all groups. Granting this permission implicitly grants view-user as well.
view-message	View messages stored in the server.
view-route	View routes.
view-server	View server configuration and information.
view-user	View any user.



Any type of modification to an item requires that the user can view that item. Therefore, granting any create, modify, delete, change, or purge permission implicitly grants the permission to view the associated item.

Granting the view permissions is useful when you want specific users to only be able to view items. It is not necessary to grant the view permission if a user already has a permission that allows the user to modify the item.

Global permissions are stored in the `acl.conf` file, along with all other permissions. Global permissions in this file have the following syntax:

```
ADMIN USER=<username> PERM=<permission>
```

or

```
ADMIN GROUP=<groupname> PERM=<permission>
```

For example, if a user named BOB is granted the `view-user` global administration permission and the group `sys-admins` is granted the `change-acl` permission, the following entries are added to the `acl.conf` file:

```
ADMIN USER=BOB PERM=view-user
ADMIN GROUP=sys-admins PERM=change-acl
```

Destination-Level Permissions

Administrators can be granted permissions on each destination. Destination-level permissions control the administration functions a user can perform on a specific destination. Global permissions granted to a user override any destination-level permissions.

The typical use of destination-level administration permissions is to specify permissions on wildcard destinations for different groups of users. This allows you to specify particular destinations over which a group of users has administrative control. For example, you may allow one group to control all `ACCOUNTING.* topics`, and another group to control all `PAYROLL.* queues`.

Table 34 describes the destination-level administration permissions.

Table 34 Destination-level administration permissions

Permission	Allows Administrator To...
view	View information for this destination.
create	Create the specified destination. This permission is useful when used with wildcard destination names. This allows the user to create any destination that matches the specified parent.
delete	Delete this destination.
modify	Change the properties for this destination.
purge	Either purge this queue, if the destination is a queue, or purge the durable subscribers, if the destination is a topic with durable subscriptions.



Any type of modification to an item requires that the user can view that item. Therefore, granting create, modify, delete, change, or purge implicitly grants the permission to view the associated item.

Granting the view permissions is useful when you want specific users to only be able to view items. It is not necessary to grant the view permission if a user already has a permission that allows the user to modify the item.

Administration permissions for a destination are stored alongside all other permissions for the destination in the `acl.conf` file. For example, if user BOB has publish and subscribe permissions on topic `foo`, and then BOB is granted view permission, the acl listing would look like the following:

```
TOPIC=foo USER=BOB PERM=publish,subscribe,view
```



Both user and administrator permissions for a destination are stored in the same entry in the `acl.conf` file. This is for convenience rather than for clarity. User permissions specify the actions a client application can perform on a destination (publish, subscribe, send, receive, and so on). Administrator permissions specify what administrative commands the user can perform on the destination when using the administration tool or API.

Protection Permissions

Protection permissions allow you to group users into administrative domains so that administrators can only perform actions within their domain. An administrator can only perform administrative operations on a user that has the same protection permission as the user. There are four protection permissions (`protect1`, `protect2`, `protect3`, and `protect4`) that allow you to create four groups of administrators. Protection permissions do not apply to the `admin` user or users in the `$admin` group — these users can perform any action on any user regardless of protection permissions.

To use protection permissions, grant one of the protection permissions to a set of users (either individually, or to a defined group(s)). Then, grant the same protection permission to the administrator that can perform actions on those users.

For example, there are four departments in a company: sales, finance, manufacturing, and system administrators. Each of these departments has a defined group and a set of users assigned to the group. Within the system administrators, there is one manager and three other administrators, each responsible for administering the resources of the other departments. The manager of the system administrators can perform any administrator action. Each of the other system administrators can only perform actions on members of the groups for which they are responsible.

The user name of the manager is `mgr`, the user names of the other system administrators are `admin1`, `admin2`, and `admin3`. The following commands illustrate the grants necessary for creating the example administration structure.

```
add member $admin mgr
grant admin sales protect1
grant admin admin1 protect1,all
grant admin manufacturing protect2
grant admin admin2 protect2,all
grant admin finance protect3
grant admin admin3 protect3,all
```



You can grant a protection permission, in addition to the `all` permission. This signifies that the user has all administrator privileges for anyone who also has the same protection permission. However, if you revoke the `all` permission from a user, all permissions, including any protection permissions are removed from the access control list for the user.

An administrator is able to view users that have a different protection permission set, but the administrator can only perform actions on users with the same protection permission.

For example, `admin1` can perform any action on any user in the `sales` group, and can view any users in the `manufacturing` or `finance` groups. However, `admin1` is not able to grant permissions, change passwords, delete users from, or perform any other administrative action on users of the `manufacturing` or `finance` groups. The `mgr` user is able to perform any action on any user, regardless of their protection permission because `mgr` is a member of the `$admin` group.

Chapter 10 **Monitoring Server Activity**

System administrators must monitor and manage the TIBCO Enterprise Message Service server. The logging, monitoring, and statistics facilities provided by the server allow system administrators to effectively view system activity and track system performance.

Topics

- *Log Files and Tracing, page 226*
- *Message Tracing, page 231*
- *Monitoring Server Events, page 233*
- *Working with Server Statistics, page 238*

Log Files and Tracing

You can configure the TIBCO Enterprise Message Service server to write a variety of information to the log. Several parameters and commands control where the log is located as well as what information is written to the log. The log can be written to a file, to the system console, or to both.

Configuring the Log File

The `logfile` configuration parameter in `tibemsd.conf` controls the location and the name of the log file.

You can specify that the log file should be backed up and emptied after it reaches a maximum size. This allows you to rotate the log file and ensure that the log file does not grow boundlessly. The `logfile_max_size` configuration parameter allows you to specify the maximum size of the current log file. Set the parameter to 0 to specify no limit. Use KB, MB, or GB units.

Once the log file reaches its maximum size, it is copied to a file with the same name as the current log file except a sequence number is appended to the name of the backup file. The server queries the directory and determines the first available sequence number. For example, if the current log file is named `tibems.log`, the first copy is named `tibems.log.1`, the second is named `tibems.log.2`, and so on. You can move the files out of the log directory, if desired, and the next log file is determined based on the first available numbered backup in the log file directory.



When you remove or move log files, it is recommended that you remove or move all log files in the log file directory. The server can then restart its log file sequence with 1.

You can also dynamically force the log file to be backed up and truncated using the `rotatelog` command in `tibemsadmin`. See Command Listing on page 167 for more information about the `rotatelog` command.

For other configuration parameters that affect the log file, see Tracing and Log File Parameters on page 132.

Tracing on the Server

The TIBCO Enterprise Message Service server can be configured to produce tracing messages. These messages can describe actions performed for various areas of functionality (for example, Access Control, Administration, or Routing). These messages can also provide information about activities performed on or by the server, or the messages can provide warnings in the event of failures or illegal actions.

Trace messages can be sent to a log file, the console, or both. You configure tracing in the following ways:

- By configuring the `log_trace` and/or `console_trace` parameters in the `tibemsd.conf` file; see Table 20 on page 178.
- By specifying the `-trace` option when starting the server
- By using the `set server` command when the server is running.

`log_trace` and `console_trace` can be configured independently or together. You can configure different types of messages to go to the log file and to the console, if desired.



When you want trace messages to be sent to a log file, you must also configure the `logfile` configuration parameter. If you specify `log_trace`, and the `logfile` configuration parameter is not set to a valid file, the tracing options are stored, but they are not used until the server is started with a valid log file.

When configuring log or console tracing, you have a variety of options for the types of trace messages that can be generated. Table 35 describes the available tracing options.

Table 35 Server tracing options (Sheet 1 of 2)

Trace Option	Description
DEFAULT	Sets the trace options to the default set. This includes: <ul style="list-style-type: none">• INFO• WARNING• ACL• LIMITS• ROUTE• ADMIN• RVADV• CONNET_ERROR• CONFIG• MSG
INFO	Prints messages as the server performs various internal housekeeping functions, such as creating a configuration file, opening the persistent database files, and purging messages. Also prints a message when tracking by message ID is enabled or disabled.
WARNING	Prints a message when a failure of some sort occurs, usually because the user attempts to do something illegal. For example, a message is printed when a user attempts to publish to a wildcard destination name.
ADMIN	Prints a message whenever an administration function is performed.
LIMITS	Prints a message when a limit is exceeded, such as the maximum size for a destination.
ACL	Prints a message when a user attempts to perform an unauthorized action. For example, if the user attempts to publish a message to a secure topic for which the user has not been granted the publish permission.

Table 35 Server tracing options (Sheet 2 of 2)

Trace Option	Description
SSL	Prints detailed messages of the SSL process, including certificate content.
SSL_DEBUG	Prints messages that trace the establishment of SSL connections.
ROUTE	Prints a message when routes are created or when a route connection is established.
ROUTE_DEBUG	Prints a message for each message that is sent over a route.
CONNECT	Prints a message when a user attempts to connect to the server.
CONNECT_ERROR	Prints a message when an error occurs on a connection.
PRODCONS	Prints a message when a client creates or closes a producer or consumer.
DEST	Prints a message when a dynamic destination is created.
TX	Prints a message when a client performs a transaction.
LDAP_DEBUG	Prints messages when LDAP is used for authentication or to obtain group information.
AUTH	Prints a message when the server authenticates a user using an external LDAP system.
MSG	Specifies that message trace messages should be printed. Message tracing is enabled/disabled on a destination or on an individual message. If message tracing is not enabled for any messages or destinations, no trace messages are printed when this option is specified for log or console tracing. See Message Tracing on page 231 for more information about message tracing.
FLOW	Prints a message when the server enforces flow control or stops enforcing flow control on a destination.
RVADV	Prints TIBCO Rendezvous advisory messages whenever they are received.

Specify tracing with a comma-separated list of trace options. You may specify trace options in three forms:

- **plain** A trace option without a prefix character replaces any existing trace options.
- **+** A trace option preceded by + adds the option to the current set of trace options.
- **-** A trace option preceded by - removes the option from the current set of trace options.

Examples

The following example sets the trace log to only show messages about access control violations.

```
log_trace=ACL
```

The next example sets the trace log to show all default trace messages, in addition to SSL messages, but ADMIN messages are not shown.

```
log_trace=DEFAULT, -ADMIN, +SSL
```

The next example sends a trace message to the console when a TIBCO Rendezvous advisory message arrives.

```
console_trace=RVADV
```

Message Tracing

In addition to other server activity, you can trace messages as they are processed. Trace entries for messages are only generated for destinations or messages that specify tracing should be performed. For destinations, you specify the `trace` property to enable the generation of trace messages. For individual messages, the `JMS_TIBCO_MSG_TRACE` property specifies that tracing should be performed for this message, regardless of the destination settings. The sections below describe the tracing properties for destinations and messages.

Message trace entries can be output to either the console or the log. The `MSG` trace option specifies that message trace entries should be displayed, and the `DEFAULT` trace option includes the `MSG` option. See [Tracing on the Server](#) on page 227 for more information about specifying trace options.

You must set the tracing property on either destinations or messages and also set the `MSG` or `DEFAULT` trace option on the console or the log before you can view trace entries for messages.



EMS tracing features do not filter unprintable characters from trace output. If your application uses unprintable characters within messages (whether in data or headers), the results of message tracing are unpredictable.

Enabling Message Tracing for a Destination

The `trace` property on a destination specifies that trace entries are generated for that destination. This property can optionally be specified as `trace=body`. Setting `trace=body` includes the message body in trace messages. Setting `trace` without the `body` option specifies that only the message sequence and message ID are included in the trace message.

When message tracing is enabled for a destination, a trace entry is output for each of the following events that occur in message processing:

- messages are received into a destination
- messages are sent to consumers
- messages are imported or exported to/from an external system
- messages are acknowledged
- messages are sent across a destination bridge
- messages are routed

Also, any reply messages are traced when the request message is sent to a destination that has the `trace` property. In the case of exported messages, when a message is sent to a destination that has a `trace` property, the reply message automatically generates a trace entry when `JMSReplyTo` is set to a temporary destination. If the reply to an exported message is sent to a static destination, to generate a trace entry, the reply destination must have the `trace` property set.

Enabling Message Tracing on a Message

You can enable tracing on individual messages by setting the `JMS_TIBCO_MSG_TRACE` property on the message. The value of the property can be `null` or `body`. Setting the property to `null` specifies only the message ID and message sequence will be included in the trace entries for the message. Setting the property to `body` specifies the message body will be included in the trace entries for the message.

When the `JMS_TIBCO_MSG_TRACE` property is set for a message, trace entries are generated for the message as it is processed, regardless of whether the `trace` property is set for any destinations the message passes through. Trace messages are generated for the message when it is sent by the producer and when it is received by the consumer.

Monitoring Server Events

The TIBCO Enterprise Message Service server can publish topic messages for several system events. For example, the server can publish a message when users connect or disconnect. System event messages contain detail about the event stored in properties of the message. This section gives an overview of the monitoring facilities provided by the server. For a list of monitor topics and a description of the message properties for each topic, see Appendix C, Monitor Messages, on page 345.

System Monitor Topics

The TIBCO Enterprise Message Service server can publish messages to various topics when certain events occur. There are several types of event classes, each class groups a set of related events. For example, some event classes are connection, admin, and route. Each event class is further subdivided into the events for each class. For example, the connection class has two events: connect and disconnect. These event classes are used to group the system events into meaningful categories.

All system event topic names begin with `$sys.monitor.` The remainder of the name is the event class followed by the event. For example, the server publishes a message to the topic `$sys.monitor.connection.disconnect` whenever a client disconnects from the server. The naming scheme for system event topics allows you to create wildcard subscriptions for all events of a certain class. For example, to receive messages whenever clients connect or disconnect, you would create a topic subscriber for the topic `$sys.monitor.connection.*`.

Monitor topics are created and maintained by the server. There is no need to explicitly create monitor topics in the configuration files or using the administration tool or API.

Monitoring Messages

You can monitor messages processed by a destination as they are sent, received, or acknowledged. The `$sys.monitor` topic for monitoring messages has the following format:

```
$sys.monitor.<D>.<E>.<destinationName>
```

Where *D* is the type of destination, *E* is the event you wish to monitor, and *destinationName* is the name of the destination whose messages you wish to monitor. Table 36 describes the possible values of *D* and *E* in message monitoring topics.

Table 36 Message monitoring qualifiers (Sheet 1 of 2)

Qualifier	Value	Description
D	T	Destination to monitor is a topic. Include the message body in the monitor message as a byte array. Use the <code>createFromBytes()</code> method when viewing the monitor message to recreate the message body, if desired.
	t	Destination to monitor is a topic. Do not include the message body in the monitor message.
	Q	Destination to monitor is a queue. Include the message body in the monitor message as a byte array. Use the <code>createFromBytes()</code> method when viewing the monitor message to recreate the message body, if desired.
	q	Destination to monitor is a queue. Do not include the message body in the monitor message.

Table 36 Message monitoring qualifiers (Sheet 2 of 2)

Qualifier	Value	Description
<i>E</i>	<i>s</i>	Monitor message is generated when a message is sent by the server to: <ul style="list-style-type: none"> • a consumer • a route • an external system by way of a transport
	<i>r</i>	Monitor message is generated when a message is received by the specified destination. This occurs when the message is: <ul style="list-style-type: none"> • Sent by a producer • Sent by a route • Forwarded from another destination by way of a bridge • Imported from transport to an external system
	<i>a</i>	Monitor message is generated when a message is acknowledged.
	<i>*</i>	Monitor message is generated when a message is sent, received, or acknowledged for the specified destination.

For example, `$sys.monitor.T.r.corp.News` is the topic for monitoring any received messages to the topic named `corp.News`. The message body of any received message is included in monitor messages on this topic. The topic `$sys.monitor.q.*.corp.*` monitors all message events (send, receive, acknowledge) for all queues matching the name `corp.*`. The message body is not included in this topic's messages.

The messages sent to this type of monitor topic include a description of the event, information about where the message came from (a producer, route, external system, and so on), and optionally the message body, depending upon the value of *D*. See Appendix C, Monitor Messages, on page 345 for a complete description of the properties available in monitoring messages.

You must explicitly subscribe to a message monitoring topic. That is, subscribing to `$sys.monitor.>` will subscribe to all topics beginning with `$sys.monitor`, but it does not subscribe you to any specific message monitoring topic such as `$sys.monitor.T.*.foo.bar`. You can, however, specify wildcards in the *destinationName* portion of the message monitoring topic to subscribe to the

message monitoring topic for all matching destinations. For example, you can subscribe to `$sys.monitor.T.r.>` to monitor all messages received by all topics. For performance reasons, you may want to avoid subscribing to too many message monitoring topics. See *Performance Implications of Monitor Topics* on page 237 for more information.

Viewing Monitor Topics

Monitor topics are just like any other topic. To view these topics, create a client application that subscribes to the desired topics.

Because monitor topics contain potentially sensitive system information, authentication and permissions are always checked when clients access a monitor topic. That is, even if authentication for the server is disabled, clients are not able to access monitor topics unless they have logged in with a valid username and password and the user has permission to view the desired topic.

The `admin` user and members of the `$admin` group have permission to perform any server action, including subscribing to monitor topics. All other users must be explicitly granted permission to view monitor topics before the user can successfully create subscribers for monitor topics. For example, if user `BOB` is not a member of the `$admin` group, and you wish to allow user `BOB` to monitor all connection events, you can grant `BOB` the required permission with the following command using the administration tool:

```
grant topic $sys.monitor.connection.* BOB subscribe
```

Bob's application can then create a topic subscriber for `$sys.monitor.connect.*` and view any connect or disconnect events.



Topics starting with `$sys.monitor` do not participate in any permission inheritance from parent topics other than those starting with `$sys.monitor` (that is, `*.*` or `*.>` is not a parent of `$sys.monitor`).

Therefore, granting permission to a user to subscribe to `>` does not allow that user to subscribe to `$sys.monitor` topics. You must explicitly grant users permission to `$sys.monitor` topics (or parent topics, such as `$sys.monitor.admin.*`) for a user to be able to subscribe to that topic.

Monitor topics publish messages of type `MapMessage`. Information about the event is stored within properties in the message. Each system event has different properties. Appendix C, *Monitor Messages*, on page 345 describes each of the monitor topics and the message properties for the messages published on that topic. Your application can receive and display all or part of a monitor message, just as it would handle any message sent to a topic.

Monitor messages are like any topic messages, so you can have any number of applications that subscribe to monitor messages. You can create different applications that subscribe to different monitor topics, or you can create one application that subscribes to all desired monitor topics. Your topic subscribers can also use message selectors to filter the monitor messages so your application receives only the messages it is interested in.

Performance Implications of Monitor Topics

The TIBCO Enterprise Message Service server only generates messages for monitor topics that currently have subscribers. So, if no applications subscribe to monitor topics, no monitor messages are generated. Generating a monitor message does consume system resources, and therefore you should consider what kinds of monitoring your environment requires. System performance is affected by the number of subscribers for monitor topics as well as the frequency of messages for those topics.

For development and testing systems, monitoring all system events is probably desirable. Usually, development and testing systems do not have large message volumes, and monitoring can give you information about system problems.

For production systems, monitoring all events may have an adverse effect on system performance. Therefore, you should not create topic subscribers for `$sys.monitor.>` in your production system. Also, monitor events are likely to be added in future releases, so the number of monitor topics may grow. Subscriptions to monitor topics in production systems should always be limited to specific monitor topics or wildcard subscriptions to specific classes of monitor topics that are required.

Also, consider the frequency of messages to each monitor topic. System administration events, such as creating topics, routes, and changing permissions, do not occur frequently, so creating subscriptions for these types of events will most likely not have a significant effect on performance.

Also, using message selectors to limit monitor messages can improve performance slightly. The server does not send any messages that do not match a subscriber's message selector. Even though the message is not sent, the message is still generated. Therefore there is still system overhead for subscribers to a monitor topic, even if all messages for that topic do not match any subscriber's message selector filter.

Working with Server Statistics

The TIBCO Enterprise Message Service server allows you to track incoming and outgoing message volume, message size, and message statistics for the server overall as well as for each producer, consumer, or route. You can configure the type of statistics collected, the interval for computing averages, and amount of detail for each type.

Statistic tracking can be set in the server's configuration file, or you can change the configuration dynamically using commands in the administration tool or by creating your own application with the administration APIs.

Statistics can be viewed using the administration tool, or you can create your own application that performs more advanced analysis of statistics using the administration APIs.

This section details how to configure and view statistics using the configuration files and administration tool commands. For more information about the administration APIs, see the description of `com.tibco.tibjms.admin` in the online documentation.



The TIBCO Enterprise Message Service server tracks the number of incoming or outgoing messages, but only messages sent or received by a producer, consumer, or route are tracked. The server also sends system messages, but these are not included in the number of messages.

However, the server can add a small amount of data to a message for internal use by the server. This overhead is counted in the total message size, and you may notice that some messages have a greater message size than expected.

Overall Server Statistics

The server always collects certain overall server statistics. This includes the rate of inbound and outbound messages (expressed as number of messages per second), message memory usage, disk storage usage, and the number of destinations, connections, and durable subscriptions. Gathering this information consumes virtually no system resources, therefore these statistics are always available. You can view overall server statistics by executing either the `show server` or `show config` commands.

The default interval for collecting overall server statistics is 1 second. You may wish to view average system usage statistics over a larger interval. The `server_rate_interval` configuration parameter controls the collection interval for server statistics. The parameter can be set in the configuration file or dynamically using the `set server` command. This parameter can only be set to positive integers greater than zero.

Enabling Statistic Gathering

Each producer, consumer, destination, and route can gather overall statistics and statistics for each of its destinations. To enable statistic gathering, you must set the `statistics` parameter to `enabled`. This parameter can be specified in the configuration file, and it can be changed dynamically using the `set server` command.

The `statistics` parameter allows you to globally enable and disable statistic gathering. Statistics are kept in server memory for the life of each object. If you wish to reset the total statistics for all objects to zero, disable statistic gathering, then re-enable it. Server statistics are also reset when the server shuts down and restarts, or in the event of a fault-tolerant failover.

For each producer, consumer, destination, and route the total number of sent/received messages and total size of messages is maintained. Also, producers and consumers keep these statistics for each destination that they use to send or receive messages.

The rate of incoming/outgoing messages and message size is calculated over an interval. By default, the average is calculated every 3 seconds. You can increase or decrease this value by altering the `rate_interval` parameter. This parameter can be set in the configuration file or dynamically using the `set server` command. Setting this parameter to 0 disables the tracking of statistics over an interval—only the total statistics for the destination, route, producer, or consumer are kept.

Gathering total statistics for producers, consumers, destinations, and routes consumes few system resources. Under most circumstances, enabling statistic gathering and average calculations should not affect system performance.

Detailed Statistics

In some situations, the default statistic gathering may not be sufficient. For example, if a topic subscriber subscribes to wildcard topics, the total statistics for all topics that match the wildcard are kept. You may wish to get further detail in this case and track the statistics for each actual topic the subscriber receives.

The following situations may require detailed statistic gathering:

- Topic subscribers that subscribe to wildcard topics

- Message producers that do not specify a destination when they are created. These message producers can produce messages for any destination, and the destination name is specified when a message is sent.
- Routes can have incoming and outgoing messages on many different topics.

To enable detailed statistics, set the `detailed_statistics` parameter to the type of statistics you wish to receive. The parameter can have the following values:

- `NONE` — disables detailed statistic gathering.
- `CONSUMERS` — enables detailed statistics for topic subscribers with wildcard topic names.
- `PRODUCERS` — enables detailed statistics for producers that do not specify a destination when they are created.
- `ROUTES` — enables detailed statistics for routes

You can set the `detailed_statistics` parameter to `NONE` or any combination of `CONSUMERS`, `PRODUCERS`, or `ROUTES`. To specify more than one type of detailed statistic gathering, provide a comma-separated list of values. You can set the `detailed_statistics` parameter in the configuration file or dynamically by using the `set server` command. For example, the following `set server` command enables detailed statistic tracking for producers and routes.

```
set server detailed_statistics = PRODUCERS,ROUTES
```

Collecting detailed statistics does consume memory, and can adversely affect performance when gathering a high volume of statistics. There are two parameters that allow you to control resource consumption when collecting detailed statistics. First, you can control the amount of time statistics are kept, and second you can set a maximum amount of memory for detailed statistic gathering. When application programs create many dynamic destinations, we recommend against gathering detailed statistics.

The `statistics_cleanup_interval` parameter controls how long detailed statistics are kept. This parameter can be set either in the configuration file or dynamically with the `set server` command. By default, statistics are kept for 15 seconds. For example, if there is a topic subscriber for the topic `foo.*`, and the subscriber receives a message on topic `foo.bar`, if no new messages arrive for topic `foo.bar` within 15 seconds, statistics for topic `foo.bar` are deleted for that consumer. You can set this parameter to 0 to signify that all detailed statistics are to be kept indefinitely. Of course, statistics for an object only exist as long as the object itself exists. That is, if a message consumer terminates, all detailed statistics for that consumer are deleted from memory.

The `max_stat_memory` parameter controls the amount of memory used by detailed statistics. This parameter can be set either in the configuration file or dynamically with the `set server` command. By default, this parameter is set to 0 which signifies that detailed statistics have no memory limit. If no units are specified, the value of this parameter is in bytes. Optionally, you can specify units as KB, MB, or GB. When the specified limit is reached, the server stops collecting new statistics. The server will only resume collecting statistics if the amount of memory used decreases (for example, if the `statistics_cleanup_interval` is set and old statistics are removed).

Displaying Statistics

When statistic collecting is enabled, you can view statistics for producers, consumers, routes, and destinations using the `show stat` command in the administration tool.

The `show stat` command allows you to filter the statistics based on destination name, user name, connection ID, or any combination of criteria. You can optionally specify the `total` keyword to retrieve only the total statistics (this suppresses the detailed output). You can also optionally specify the "wide" keyword when displaying statistics for destinations or routes. This specifies that inbound and outbound message statistics should be displayed on the same line (the line can be 100 characters or more).

The following illustrates displaying statistics for a route where detailed statistic tracking is enabled.

```
tcp://server1:7322> show stat route B
Inbound statistics for route 'B':
```

Destination	Total Count		Rate/Second	
	Msgs	Size	Msgs	Size
<total>	189	37.9 Kb	10	2.0 Kb
Topic: dynamic.0	38	7.6 Kb	2	0.4 Kb
Topic: dynamic.1	38	7.6 Kb	2	0.4 Kb
Topic: dynamic.2	38	7.6 Kb	2	0.4 Kb
Topic: dynamic.3	38	7.6 Kb	2	0.4 Kb
Topic: dynamic.4	37	7.4 Kb	2	0.4 Kb

```
Outbound statistics for route 'B':
```

Destination	Total Count		Rate/Second	
	Msgs	Size	Msgs	Size
<total>	9538	1.9 MB	10	2.1 Kb
Topic: dynamic.0	1909	394.9 Kb	2	0.4 Kb
Topic: dynamic.1	1908	394.7 Kb	2	0.4 Kb
Topic: dynamic.2	1907	394.5 Kb	2	0.4 Kb
Topic: dynamic.3	1907	394.5 Kb	2	0.4 Kb
Topic: dynamic.4	1907	394.5 Kb	2	0.5 Kb

See `show stat` on page 194 for more information and detailed syntax of the `show stat` command.

Chapter 11 **Deploying the Application**

This chapter describes deploying the TIBCO Enterprise Message Service application.

Topics

- *Running the Server, page 244*
- *Security Considerations, page 248*
- *Running TIBCO Enterprise Message Service Client-Side Application, page 252*
- *Connecting Directly to TIBCO Enterprise Message Service Server, page 253*
- *Using JNDI with TIBCO Enterprise Message Service, page 255*

Running the Server

In order to use TIBCO Enterprise Message Service with your applications, the TIBCO Enterprise Message Service Server must be running. The server and the clients work together to implement TIBCO Enterprise Message Service. The server implements all types of message persistence; no messages are stored on the client side.

Starting the Server

TIBCO Enterprise Message Service Server is located in the bin subdirectory of the TIBCO Enterprise Message Service installation directory.

- 1. Navigate to the bin subdirectory
- 2. Type `tibemsd [options]`

where options are described in Table 37. The command options to `tibemsd` are similar to the parameters you specify in `tibemsd.conf`, and the command options override any value specified in the parameters. See Table 18 on page 120 for more information about configuration parameters and more information about each parameter.

Table 37 *tibemsd Options (Sheet 1 of 2)*

Option	Description
<code>-config <config file name></code>	<code><config file name></code> is the name of the main configuration file for <code>tibemsd</code> server. Default is <code>tibemsd.conf</code> .
<code>-trace <items></code>	Specifies the trace items. These items are not stored in the configuration file. The value has the same format as the value of <code>log_trace</code> parameter specified with <code>set server</code> command of the administration tool; see Tracing on the Server on page 227.
<code>-ssl_password <string></code>	Private key password.
<code>-ssl_trace</code>	Print the certificates loaded by the server and do more detailed tracing of SSL-related situation.
<code>-ssl_debug_trace</code>	Turns on tracing of SSL connections.
<code>-ft_active <active_url></code>	URL of the active server. If this server can connect to the active server, it will act as a backup server. If this server cannot connect to the active server, it will become the active server.


Table 37 *tibemsd Options (Sheet 2 of 2)*

Option	Description
<code>-ft_heartbeat<seconds></code>	Heartbeat signal for the active server, in seconds. Default is 3.
<code>-ft_activation <seconds></code>	Activation interval (maximum length of time between heartbeat signals) which indicates that active server has failed. Set in seconds: default is 10. This interval should be set to at least twice the heartbeat interval.

emsntsreg

Utility

- Purpose

Register or unregister the EMS server daemon as a Windows service.
- 

This utility applies only to Microsoft **Windows** (all supported versions, including NT, 2000 and XP).
- Syntax

```
emsntsreg /i [/a] service_name directory [arguments] [suffix]
emsntsreg /r [service_name] [suffix]
```
- Remarks

Some situations require the EMS server to start automatically. You can satisfy this requirement by registering the server daemon with the Windows service manager. This utility facilitates registry.
- Restrictions

You must have administrator privileges to change the Windows registry.

This command does not support space characters in directory paths or file names.
- Location

Locate this utility program as an executable file in the EMS bin directory.

Parameter	Description
/i	Insert a new service in the registry (that is, register a new service).
/a	Automatically start the new service. Optional with /i.
service_name	<p>Insert or remove a service with this base name.</p> <p>When inserting a service, this parameter is required, and must be tibemsd.</p> <p>When removing a service, this parameter is optional. However, if it is present, it must be tibemsd.</p>
directory	Use this directory pathname to locate the tibemsd executable. Required.
arguments	<p>Supply command line arguments to tibemsd. Optional with /i.</p> <p>Enclose the entire arguments string in double quote characters.</p>
suffix	When registering more than one tibemsd service, you can use this suffix to distinguish between them in the Windows services applet. Optional.
/r	Remove a service from the registry.

Register

To register tibemsd as a Windows service, run the utility with this command line:

```
emsntsreg /i [/a] tibemsd directory [arguments] [suffix]
```

Example 1 This simple example registers one service:

```
emsntsreg /i tibemsd C:\tibco\ems\bin
```

Example 2 This example registers a service with command line arguments:

```
emsntsreg /i tibemsd C:\tibco\ems\bin "-trace DEFAULT"
```

Example 3 This pair of example commands registers two services with different configuration files. In this example, the numerical suffix and the configuration directory both reflect the port number that the service uses.

```
emsntsreg /i tibemsd C:\tibco\ems\bin "-config C:\tibco\ems\7222\tibemsd.conf" 7222
```

```
emsntsreg /i tibemsd C:\tibco\ems\bin "-config C:\tibco\ems\7223\tibemsd.conf" 7223
```

Notice these aspects of this example:

- When you supply a `-config` argument, the service process finds the directory containing the main configuration file (`tibemsd.conf`), and creates all secondary configuration files in that directory. In this example, each service uses a different configuration directory.
- When you register several EMS services, you must avoid configuration conflicts. For example, two instances of `tibemsd` cannot listen on the same port.

Remove To unregister a service, run the utility with this command line:

```
emsntsreg /r [service_name] [suffix]
```

Both parameters are optional. If the *service_name* is present, it must be `tibemsd`. To supply the *suffix* parameter, you must also supply the *service_name*. When both parameters are absent, the utility removes the service named `tibemsd`.

Command Summary To view a command line summary, run the utility with this command line:

```
emsntsreg
```

Windows Services Applet The Windows services applet displays the name of each registered service. For EMS services, it also displays this additional information:

- The suffix (if you supply one)
- The process ID (PID)—when the service is running

Security Considerations



This section highlights information relevant to secure deployment. We recommend that all administrators read this section.

Secure Environment

To ensure secure deployment, EMS administration must meet the following criteria:

- **Correct Installation** EMS is correctly installed and configured.
- **Physical Controls** The computers where EMS is installed are located in areas where physical entry is controlled to prevent unauthorized access. Only authorized administrators have access, and they cooperate in a benign environment.
- **Domain Control** The operating system, file system and network protocols ensure domain separation for EMS, to prevent unauthorized access to the server, its configuration files, LDAP servers, etc.
- **Benign Environment** Only authorized administrators have physical access or domain access, and those administrators cooperate in a benign environment.

Destination Security

Three interacting factors affect the security of destinations (that is, topics and queues). In a secure deployment, you must properly configure all three of these items:

- The server's authorization parameter (see *Authorization Parameter*, below)
- The secure property of individual destinations (see *secure* on page 36)
- The ACL permissions that apply to individual destinations (see *Authentication and Permissions* on page 199)

Authorization Parameter

The server's `authorization` parameter acts as a master switch for checking permissions for connection requests and operations on secure destinations. The default value of this parameter is `disabled`—the server does not check any permissions, and allows all operations. For secure deployment, you must enable this parameter.

Admin Password

For ease in installation and initial testing, the default setting for the `admin` password is no password at all. Until you set an actual password, the user `admin` can connect without a password. Once the administrator password has been set, the server always requires it.

To configure a secure deployment, the administrator must change the `admin` password immediately after installation; see [Assign a Password to the Administrator](#) on page 165.

Connection Security

When `authorization` is enabled, the server requires a name and password before users can connect. Only authenticated users can connect to the server. The form of authentication can be either an X.509 certificate or a username and password (or both).

When `authorization` is disabled, the server does not check user authentication; all user connections are allowed. However, even when `authorization` is disabled, the user `admin` must still supply the correct password to connect to the server.

Even when `authorization` is enabled, the administrator (`admin`) may explicitly allow anonymous user connections, which do not require password authorization. To allow these connections, create a user with the name `anonymous` and no password.



Creating the user `anonymous` does not mean that `anonymous` has all permissions. Individual topics and queues can still be secure, and the ability to use these destinations (either sending or receiving) is controlled by the access control list of permissions for those destinations. The user `anonymous` can access only non-secure destinations.

Nonetheless, this feature (anonymous user connections) is outside the tested configuration of EMS security certification.

For more information on destination security, refer to the [destination property secure](#) on page 36, and [Adding the secure Property to the Topic](#) on page 330.

Communication Security

For communication security between servers and clients, and between servers and other servers, you must explicitly configure SSL within EMS; see [Using the SSL Protocol](#) on page 265.

SSL communication requires software to implement SSL on both server and client. The EMS server includes the OpenSSL implementation. Java client programs must use either JSSE (part of the Java environment) or separately purchased SSL software from Entrust; neither of these are part of the EMS product. C client programs can use the OpenSSL library shipped with EMS.

Sources of Authentication Data

The server uses only one source of X.509 certificate authentication data, namely, the server parameter `ssl_server_trusted` (its value is set in EMS an configuration file). See `ssl_server_trusted` on page 141.

The server can use two sources of secure password authentication data:

- Local data from the EMS configuration files.
- External data from and LDAP.

You must safeguard the security of EMS configuration files and LDAP servers.

Timestamp

The administration tool can either include or omit a timestamp associated with the output of each command. To ensure a secure deployment, you must explicitly enable the timestamp feature. Use the following administration tool command:

```
time on
```

Passwords



Passwords are a significant point of vulnerability for any enterprise. We recommend enforcing strong standards for passwords.

For security equivalent to single DES (an industry minimum), security experts recommend passwords that contain 8–14 characters, with at least one upper case character, at least one numeric character, and at least one punctuation character.

EMS software does not automatically enforce such standards for passwords. You must enforce such policies within your organization.

Audit Trace Logs

Audit information is output to log files (and `stderr`), and is configured by the server parameters `log_trace` and `console_trace` (see Tracing and Log File Parameters on page 132).

The `DEFAULT` setting includes `+ADMIN`, so all administrative operations produce audit output. For further details, see Table 35, Server tracing options, on page 228.

Audit information in log files is always timestamped.

Administrators can read and print the log files for audit review using tools (such as text editors) commonly available within all IT environments. EMS software does not include a special tool for audit review.

Running TIBCO Enterprise Message Service Client-Side Application

In order to run the client-side application, you must implement the items on the programmer's check list, and then connect the client side to the server. There are two methods to connect the client to the server: directly and with the JNDI interface. This section describes the programmer's checklist, security considerations, and both methods of connecting to the server.

Programmer's Checklist

In order to compile and run a client-side Java application using TIBCO Enterprise Message Service, you must be sure:

1. Your application imports the following packages:

```
import javax.jms.*;  
import javax.naming.*;
```

2. The CLASSPATH includes the following jar files:

```
jms.jar  
jndi.jar  
tibjms.jar
```

3. If SSL is used for communication, add the following files to the CLASSPATH:

```
tibcrypt.jar  
jnet.jar  
jcert.jar  
jsse.jar
```

All necessary jar files are located in the java subdirectory of the TIBCO Enterprise Message Service installation directory.

Connecting Directly to TIBCO Enterprise Message Service Server

Client applications must connect to a running instance of TIBCO Enterprise Message Service server in order to perform any JMS operations.

Normally client applications use JNDI access to look up a `ConnectionFactory` object in order to connect to the server. For more information on JNDI-based access to the server, refer to *Using JNDI with TIBCO Enterprise Message Service* on page 255.

However, in rare cases applications may connect to the server directly.

In order to connect to the server directly, the application must use one of the following statements.

For the common facility interface, use:

```
ConnectionFactory factory = new
    com.tibco.tibjms.TibjmsConnectionFactory(server_url);
```

For a topic, use:

```
TopicConnectionFactory factory = new
    com.tibco.tibjms.TibjmsTopicConnectionFactory(server_url);
```

For a queue, use:

```
QueueConnectionFactory factory = new
    com.tibco.tibjms.TibjmsQueueConnectionFactory(server_url);
```

The *server_url* parameter in these expressions is a Java string defining the protocol and the address of the running instance of the TIBCO Enterprise Message Service Server. The *server_url* parameter has the form:

protocol : // *host* : *port*



The supported *protocols* are **tcp** and **ssl**.

Examples of the server url string are:

```
tcp://jmshost:7255
ssl://localhost:7243
```

You can use short forms of the *server-url*, as described below:

- The protocol can be omitted: it defaults to **tcp**.
- The port can be omitted: it defaults to **7222**.

For example, valid forms of the server url are:

- **anotherhost:7333** (equivalent to **tcp://anotherhost:7333**)

- **mainhost** (equivalent to **tcp://mainhost:7222**)

Using JNDI with TIBCO Enterprise Message Service

In most cases, client applications use the JNDI interface to look up connection factories and Topic and Queue objects.

Dynamic Topics and Queues

TIBCO Enterprise Message Service allows dynamic creation of topics and queues using the following JMS API methods:

- `session.createTopic(<topic name>);`
- `session.createQueue(<queue name>);`
- `TopicSession.createTopic(<topic name>);`
- `QueueSession.createQueue(<queue name>);`

Since dynamic topics and queues do not appear in the configuration files, you cannot use the JNDI interface to look up these topics and queues.

For more information about connecting to the server without the JNDI interface, refer to Connecting Directly to TIBCO Enterprise Message Service Server on page 253.

Static Topics and Queues

In order to use JNDI access to TIBCO Enterprise Message Service connection factories and topics and queues, the application should use the following parameters:

- provider context factory
`com.tibco.tibjms.naming.TibjmsInitialContextFactory`
- JNDI provider URL in the form
`tibjmsnaming://host:port`
- Topics and Queues or optional JNDI names, created using the administration tool, the administration APIs, or in the configuration files.
- To obtain connection factory, topic, or queue objects, the client application should use JNDI calls:

```
Topic topic =(javax.jms.Topic)jndiContext.lookup(<topic-name>);
Queue queue =(javax.jms.Queue)jndiContext.lookup(<queue-name>);
ConnectionFactory cf =
    (javax.jms.ConnectionFactory)jndiContext.lookup(<connection-
factory-name>);
```

- If there are topics and queues with same names in the configuration files, the client application must use context qualifiers:

```
Topic topic =
(javax.jms.Topic)jndiContext.lookup("$topics:" + <topic-name>);
Queue queue =
(javax.jms.Queue)jndiContext.lookup("$queues:" + <queue-name>);
```



Formerly, the syntax for topic and queue lookup was `$topic.` and `$queue.`

This syntax is supported for backward compatibility, but the new syntax is preferred.

Using context qualifiers is not required if there are no topics and queues with the same names in the configuration files.



The provider URL **host:port** value is one of the listen ports of TIBCO Enterprise Message Service. There is no separate port defined for JNDI access.

Example

This section provides an example of accessing JMS administered objects when using TIBCO Enterprise Message Service.

```
static final String providerContextFactory =
    "com.tibco.tibjms.naming.TibjmsInitialContextFactory";

static final String defaultProviderURL =
    "tibjmsnaming://jmshost:7222";

try
{
    Hashtable env = new Hashtable();
    env.put(Context.INITIAL_CONTEXT_FACTORY, providerContextFactory);
    env.put(Context.PROVIDER_URL, providerURL);
    jndiContext = new InitialContext(env);
}
catch (NamingException e)
{
    System.out.println("Failed to create InitialContext");
    e.printStackTrace();
}

ConnectionFactory factory =
    (javax.jms.ConnectionFactory)
    jndiContext.lookup("ConnectionFactory");

TopicConnectionFactory topicFactory =
    (javax.jms.TopicConnectionFactory)
    jndiContext.lookup("TopicConnectionFactory");

QueueConnectionFactory queueFactory =
    (javax.jms.QueueConnectionFactory)
```

```
jndiContext.lookup("QueueConnectionFactory");

Topic topic = (javax.jms.Topic)jndiContext.lookup("topic.sample");
Queue queue = (javax.jms.Queue)jndiContext.lookup("queue.sample");
```

Looking Up Objects Using Full URL Names

Administered objects can also be looked up using full URL names. In this case, the `Context.PROVIDER_URL` property is not provided to the `InitialContext`. Instead, the `Context.URL_PKG_PREFIXES` property is provided using the statement:

```
env.put(Context.URL_PKG_PREFIXES, "com.tibco.tibjms.naming");
```

If using full URL names, you can look up topics or queues like the following example:

```
Topic topic = (javax.jms.Topic)jndiContext.lookup(
    "tibjmsnaming://jmshost:7222/topic.sample");
Queue queue = (javax.jms.Queue)jndiContext.lookup(
    "tibjmsnaming://jmshost:7222/queue.sample");
```

For further information on how to use JNDI access, refer to the `tibjmsJNDI.java` example included with TIBCO Enterprise Message Service.

Using SSL with JNDI Lookups

TIBCO Enterprise Message Service client programs can perform secure JNDI lookups using the Secure Sockets Layer (SSL) protocol. To accomplish this, the client program must set SSL properties in the environment when the `InitialContext` is created. The SSL properties are similar to the SSL properties for the TIBCO Enterprise Message Service server. See Chapter 12, *Using the SSL Protocol*, on page 265 for more information about using SSL in the TIBCO Enterprise Message Service server.

Table 38 describes the properties for configuring SSL when using JNDI. Refer to *TIBCO Enterprise Message Service Java API Reference* included in the online documentation for more information about using JNDI and configuring the `InitialContext`.

Table 38 SSL properties for client applications using JNDI (Sheet 1 of 6)

Property	Datatype	Description
<code>TibjmsContext.SECURITY_PROTOCOL</code>	String	The security protocol for the connection to the JNDI server. The value of this property must be <code>ssl</code> .

Table 38 SSL properties for client applications using JNDI (Sheet 2 of 6)

Property	Datatype	Description
<code>TibjmsContext.SSL_CIPHER_SUITES</code>	String	<p>Specifies the cipher suites used by the server; each suite in the list is separated by a colon (:). This parameter can use the OpenSSL name for cipher suites or the longer, more descriptive names.</p> <p>See Specifying Cipher Suites on page 281 for more information about the cipher suites available in TIBCO Enterprise Message Service and the OpenSSL names and longer names for the cipher suites.</p>
<code>TibjmsContext.SSL_DEBUG_TRACE</code>	Boolean	<p>When set to <code>true</code>, enables more detailed SSL tracing. This property is used for debugging only; it is not for use in production systems.</p>
<code>TibjmsContext.SSL_ENABLE_VERIFY_HOST</code>	Boolean	<p>Specifies whether the client should verify the server's certificate. By default, this property is set to <code>true</code>, signifying the client should verify the server's certificate.</p> <p>When this property is set to <code>false</code>, the client establishes secure communication with the server, but does not verify the server's identity.</p>

Table 38 SSL properties for client applications using JNDI (Sheet 3 of 6)

Property	Datatype	Description
<code>TibjmsContext.SSL_ENABLE_VERIFY_HOST_NAME</code>	Boolean	<p>Specifies whether the client should verify the name in the CN field of the server's certificate. By default, this property is set to <code>true</code>, signifying the client should verify the name of the connected host or the name specified in the <code>SSL_EXPECTED_HOST_NAME</code> property against the value in the server's certificate. If the names do not match, the connection is rejected.</p> <p>When this property is set to <code>false</code>, the client establishes secure communication with the server, but does not verify the server's name.</p>
<code>TibjmsContext.SSL_EXPECTED_HOST_NAME</code>	String	<p>Specifies the name the server is expected to have in the CN field of its certificate. If this parameter is not set, the expected name is the hostname of the server.</p> <p>This parameter is used when the <code>SSL_ENABLE_VERIFY_HOST_NAME</code> parameter is set to enabled.</p>
<code>TibjmsContext.SSL_HOST_NAME_VERIFIER</code>	String	<p>Constant that holds the name of SSL property specifying the custom host name verifier for JNDI lookups.</p>

Table 38 SSL properties for client applications using JNDI (Sheet 4 of 6)

Property	Datatype	Description
TibjmsContext.SSL_IDENTITY	String	<p>The client’s digital certificate.</p> <p>PEM and PSCS#12 formats allow the digital certificate to include the private key. If these formats are used and the private key is part of the digital certificate, then setting SSL_PRIVATE_KEY is optional.</p> <p>See File Names for Certificates and Keys on page 275 for more information on file types for digital certificates.</p>
TibjmsContext.SSL_IDENTITY_ENCODING	String	<p>The encoding of the client’s certificate.</p>
TibjmsContext.SSL_ISSUER_CERTIFICATES	Vector	<p>Certificate chain member for the client. Supply the entire chain, including the CA root certificate. The client reads the certificates in the chain in the order they are presented in this property.</p> <p>See File Names for Certificates and Keys on page 275 for more information on file types for digital certificates.</p>
TibjmsContext.SSL_PASSWORD	String	<p>Password for client’s private key.</p>
TibjmsContext.SSL_PRIVATE_KEY	String	<p>The name and location of the client’s private key file.</p>
TibjmsContext.SSL_PRIVATE_KEY_ENCODING	String	<p>The encoding of the client’s private key file.</p>

Table 38 SSL properties for client applications using JNDI (Sheet 5 of 6)



Property	Datatype	Description
TibjmsContext.SSL_RENEGOTIATE_INTERVAL  Obsolete Key renegotiation is deprecated in release 4.3; it is not supported in release 5.0.	Integer	<p>The client renegotiates for a new symmetric key when the cumulative size (in bytes) of the data that the client exchanges with a server reaches this threshold.</p> <p>The minimum value for this parameter is 64Kb. You can specify Kb, Mb, or Gb for the units. For example:</p> <pre>ssl_renegotiate_size = 10Gb</pre> <p>When neither of the two renegotiation parameters are set, the client does not initiate key renegotiation.</p> <p>For more information, see Renegotiating the Session Key on page 274.</p>
TibjmsContext.SSL_RENEGOTIATE_SIZE  Obsolete Key renegotiation is deprecated in release 4.3; it is not supported in release 5.0.	Integer	<p>The client renegotiates for a new symmetric key when the time (in seconds) since the last key negotiation reaches this threshold.</p> <p>The minimum value is 15 seconds. For example, you can set this parameter as follows to renegotiate every 24 hours:</p> <pre>ssl_renegotiate_interval = 86400</pre> <p>When neither of the two renegotiation parameters are set, the client does not initiate key renegotiation.</p> <p>For more information, see Renegotiating the Session Key on page 274.</p>

Table 38 SSL properties for client applications using JNDI (Sheet 6 of 6)

Property	Datatype	Description
<code>TibjmsContext.SSL_TRACE</code>	Boolean	<p>When set to <code>true</code>, enables tracing of loaded certificates.</p> <p>This prints a message to the console during startup of the server that describes each loaded certificate.</p>
<code>TibjmsContext.SSL_TRUSTED_CERTIFICATES</code>	Vector	<p>List of trusted certificates. This sets which Certificate Authority certificates should be trusted as issuers of the JNDI server certificates.</p> <p>See File Names for Certificates and Keys on page 275 for more information on file types for digital certificates.</p>
<code>TibjmsContext.SSL_VENDOR</code>	String	<p>The SSL provider name. The value of this property must be one of the following:</p> <ul style="list-style-type: none">• <code>j2se-default</code>• <code>j2se</code> (Supported only with Sun JCE/JSSE, not IBM JCE/JSSE.)• <code>entrust61</code> (The EMS Java client library works only with Entrust 7.1, and not with earlier versions. Nonetheless, the vendor constant is still <code>entrust61</code>.)

Example of Using SSL for JNDI Lookups

The following is an example of how to create an `InitialContext` that can be used to perform JNDI lookups using the SSL protocol.

```
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
        "com.tibco.tibjms.naming.TibjmsInitialContextFactory");
env.put(Context.PROVIDER_URL, "tibjmsnaming://jms host:7223");
env.put(Context.URL_PKG_PREFIXES, "com.tibco.tibjms.naming")
```

```
env.put(TibjmsContext.SECURITY_PROTOCOL, "ssl");
env.put(TibjmsContext.SSL_ENABLE_VERIFY_HOST,
        new Boolean("false"));
Context context = new InitialContext(env);
```

In this example, the port number specified for the `Context.PROVIDER_URL` is set to the SSL listen port that was specified in the server configuration file `tibjsmd.conf`. The value for `TibjmsContext.SECURITY_PROTOCOL` is set to `ssl`. Finally, the value of `TibjmsContext.ENABLE_VERIFY_HOST` is set to `"false"` to turn off server authentication. Because of this, no trusted certificates need to be provided and the client will then not verify the server it is using for the JNDI lookup against the server's certificate.

Performing Fault-Tolerant JNDI Lookups

TIBCO Enterprise Message Service can perform fault-tolerant JNDI lookups. If the primary server fails and the backup server becomes the primary, the JNDI provider automatically uses the new primary server for JNDI lookups. You accomplish this by providing multiple URLs in the `Context.PROVIDER_URL` property when creating the `InitialContext`. Specify more than one URL separated by commas (,) in the property.

Example

The following illustrates setting up the `Context.PROVIDER_URL` property with the URLs of a primary EMS server on the machine named `emshost` and a backup EMS server on the machine named `backuphost`.

```
env.put(Context.PROVIDER_URL, "tibjmsnaming://jmshost:7222,
tibjmsnaming://backuphost:7222");
```

If at any time the first EMS server fails, the JNDI provider will automatically switch to the EMS server on the host `backuphost` for JNDI lookups. If `emshost` is repaired and restarted, it then becomes the backup EMS server.

Limitations of Fault-Tolerant JNDI Lookups

Fault-tolerant JNDI lookups do not occur in the following scenarios:

- When using full URL names in argument to the lookup method.
- When looking up an object that has been bound into a foreign naming/directory service such as LDAP.

Chapter 12 Using the SSL Protocol

Secure Sockets Layer (SSL) is a protocol for transmitting encrypted data over the Internet or an internal network. SSL uses public and private key to encrypt and authenticate data transferred over the SSL connection. Most web browsers support SSL, and many Web sites and Java applications use it to obtain confidential user information, such as credit card numbers.

The SSL protocol is complex, and this chapter is not a complete description of SSL. Instead, this chapter describes how to configure SSL in the TIBCO Enterprise Message Service server and in client applications that communicate with the server. For a more complete description of SSL, see the SSL specification at <http://wp.netscape.com/eng/ssl3/>.

Topics

- *SSL Support in TIBCO Enterprise Message Service, page 266*
- *Digital Certificates, page 267*
- *Private Key Formats, page 268*
- *Overview of the SSL Protocol, page 269*
- *Renegotiating the Session Key, page 274*
- *File Names for Certificates and Keys, page 275*
- *Configuring SSL in the Server, page 276*
- *Configuring SSL in EMS Clients, page 277*
- *Specifying Cipher Suites, page 281*
- *Third-Party SSL Hardware Accelerators, page 287*

SSL Support in TIBCO Enterprise Message Service

TIBCO Enterprise Message Service supports the Secure Sockets Layer (SSL) protocol. SSL uses public and private keys to encrypt data over a network connection to secure communication between pairs of components:

- between a Java client and the `tibemsd` server
- between a C client and the `tibemsd` server
- between a COBOL client and the `tibemsd` server
- between the `tibemsadmin` tool and the `tibemsd` server
- between two routed servers
- between two fault-tolerant servers



Note that Microsoft's .NET environment does not support SSL.

Implementations

The TIBCO Enterprise Message Service server and the C client libraries use OpenSSL for SSL support. For more information, see www.openssl.org.

EMS Java clients can use either JSSE (from Sun JavaSoft) or the SSL implementation from Entrust. The EMS Java installation includes JSSE; if you prefer to use Entrust, you must purchase and install the Entrust SSL Version 7.1 implementation separately (earlier versions are not supported).

Digital Certificates

Digital certificates are data structures that represent identities. EMS uses certificates to verify the identities of servers and clients.

A digital certificate is issued either by a trusted third-party certificate authority, or by a security officer within your enterprise. Usually, each user and server on the network requires a unique digital certificate, to ensure that data is sent from and received by the correct party. A digital certificate has two parts—a public part, which identifies its owner (a user or server); and a private key, which the owner keeps confidential.

The public part of a digital certificate includes a variety of information, such as the following:

- The name of the owner, and other information required to confirm the unique identity of the subject. This information can include the URL of the web server using the digital certificate, or an email address.
- The subject's public key.
- The name of the certificate authority (CA) that issued the digital certificate.
- A serial number.
- The length of time the certificate will remain valid—defined by a start date and an end date.

The most widely-used standard for digital certificates is ITU-T X.509. TIBCO Enterprise Message Service supports digital certificates that comply with X.509 version 3 (X.509v3); most certificate authorities, such as Verisign and Entrust, comply with this standard.

Digital Certificate File Formats

TIBCO Enterprise Message Service supports the following file formats for digital certificates:

- PEM (Privacy Enhanced Mail)
- DER (Distinguished Encoding Rules)
- PKCS#7
- PKCS#12
- Java KeyStore (for client digital certificates)
- Entrust Store (for client digital certificates)

Private Key Formats

TIBCO Enterprise Message Service supports the following file formats for private keys:

- PEM (Privacy Enhanced Mail)
- DER (Distinguished Encoding Rules)
- PKCS#8
- PKCS#12

The EMS server uses OpenSSL to read private keys. It supports PEM, DER, PKCS8 and PKCS12 formats; it does *not* read Java KeyStore or Entrust Store files.

Overview of the SSL Protocol

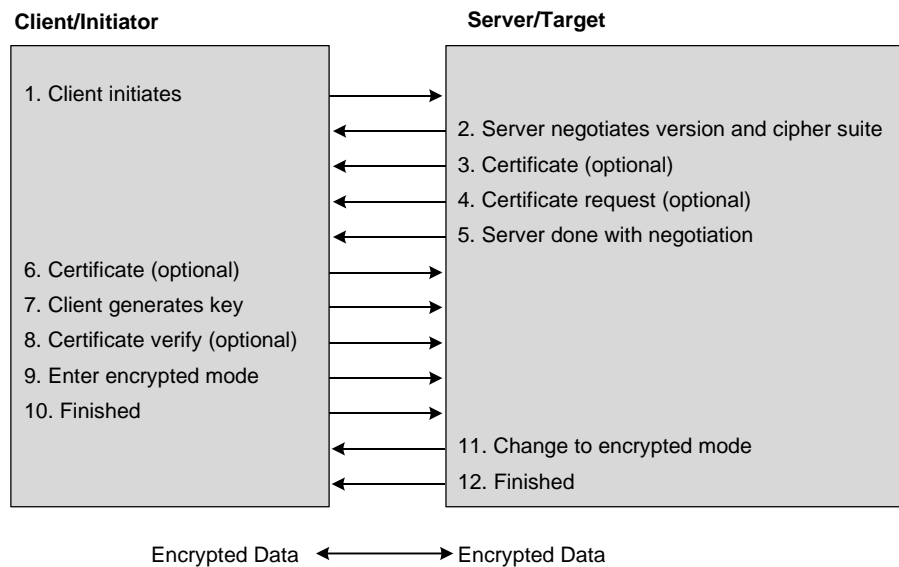


(EMS uses SSL to secure data in two situations—between a client and a server, and between two servers. To simplify the examples and explanations in this chapter, we'll focus on the interaction between client and server, although the same framework applies when two servers communicate using SSL.)

The Secure Sockets Layer (SSL) protocol involves a handshake between the initiator (in our examples, a client program) and the target (in our examples, a server).

Figure 16 illustrates the SSL protocol between the client and server.

Figure 16 SSL Protocol



When a client requests an SSL connection, the client and the server execute this handshake protocol:

1. The client sends an initiating message to the server. This message contains the highest version of SSL and the list of cipher suites the client supports. For more information about cipher suites, see *Specifying Cipher Suites* on page 281.

2. The server chooses the highest version of SSL and the best cipher suite that both the client and the server support. The server sends this information to the client.
3. If the client requires that the server authenticate itself (optional), the server sends its digital certificate or certificate chain.
4. If the server requires that the client authenticate itself (optional), the server sends a request to the client for the client's digital certificate.
5. The server then informs the client that it has completed the initial phase of the negotiation.
6. If the server requested the client's digital certificate or certificate chain (optional), the client sends it to the server.
7. The client and server then generate a symmetric encryption key. Client and server use this key to encrypt and decrypt data that they exchange.
8. If the server requires that the client authenticate itself (optional), the client sends a digital signature to the server. The server decrypts this signature using the client's public key. If the server successfully decrypts the signature, the server has authenticated the client.
9. The client informs the server that is ready to communicate secure data.
10. The client finishes the handshake protocol and can begin sending secure data.
11. The server informs the client that is ready to communicate secure data.
12. The server finishes the handshake protocol and can begin sending secure data.

When establishing an SSL connection, several of the steps described above are optional. The client and the server can specify SSL configuration parameters to control the connection steps; see *Configuring SSL in the Server* on page 276 and *Configuring SSL in EMS Clients* on page 277.

Cipher Suite Negotiation

In a fundamental segment of the SSL handshake protocol, the client and server cooperatively determine the details of encryption—including cipher algorithms, key sizes, and more. The specification of a particular set of algorithms and key size is known as a *cipher suite*. Both the client and the server can specify the cipher suites they support. During the SSL handshake, the client and server select the strongest cipher suite that they both support. For more information, see *Specifying Cipher Suites* on page 281.

Client and Server Authentication

In an optional part of the SSL handshake, the client and server authenticate themselves to each other. This step ensures that both parties are communicating with the expected applications.

Server Authentication to the Client

A rogue server could attempt to impersonate your EMS server. To prevent clients from connecting to the rogue server, the client can request that the server present its digital certificate to the client.

During the handshake, the client requests the server's digital certificate and checks its issuer against the client's list of trusted certificate authority (CA) certificates. If the CA of the server certificate is not in the client's list, the client halts communications with that server.

Client Parameters Connection factories can use the `ssl_verify_host` parameter to require that the server authenticate itself to the client. The `ssl_trusted` parameter specifies the CAs that the client trusts. For more information, see [Configuring a ConnectionFactory](#) on page 278.

EMS servers in a fault-tolerant configuration can use the `ft_ssl_verify_host` parameter to require that the other server authenticate itself. The `ft_ssl_trusted` parameter specifies the list of CAs to trust. For more information, see [Chapter 13, Fault Tolerance](#), on page 289.

Server Parameters A client may request server authentication. Therefore, the server must specify the `ssl_server_identity` parameter, and (if the password is not part of the server's digital certificate) the `ssl_password` parameter. For more information, see [SSL Server Parameters](#) on page 137.

Verifying the Server's Hostname

You can configure an EMS client to verify the server's hostname. During the handshake, the client checks the CN field of the server certificate against the name of the connected host, or if an expected hostname is provided, against that name. If the names do not match, the client rejects the connection.

Client Parameters Connection factories can use the `ssl_verify_hostname` and `ssl_expected_hostname` parameters to require verification of the hostname in the server's digital certificate. For more information, see [Configuring a ConnectionFactory](#) on page 278.

EMS servers in a fault-tolerant configuration can use the `ft_ssl_verify_hostname` and `ft_ssl_expected_hostname` parameters to require verification of the other server’s hostname. For more information, see Chapter 13, Fault Tolerance, on page 289.

Client Authentication to the Server

In many applications, the server accepts connections from any client, so server authentication is not needed. However, client passwords can be compromised, so certificate authentication is a stronger check to verify the identity of clients. To prevent rogue clients from logging into the server, the server can request that the client present its digital certificate to the server.

During the handshake, the server requests the client’s digital certificate and checks its issuer against the server’s list of trusted CAs. If the CA of the client certificate is not in the server’s list, the server halts further communications with that client.

Server Parameters	To require that clients authenticate themselves the EMS server must specify <code>true</code> for the <code>ssl_require_client_cert</code> parameter. The <code>ssl_server_trusted</code> parameter specifies the CAs the server trusts. For more information, see SSL Server Parameters on page 137.
Client Parameters	Connection factories can specify their client identities using the <code>ssl_identity</code> parameter, and (if the password is not part of the client’s digital certificate) the <code>ssl_password</code> parameter. For more information, see Configuring a ConnectionFactory on page 278.

Retrieving the Username from the Client’s Digital Certificate

The EMS server can extract the client’s username from the client’s digital certificate, and authorize it against permissions in an access control list (ACL). (Note that this type of authorization differs in intent from SSL authentication; see Chapter 9, Authentication and Permissions, on page 199.) If authorization is enabled, the server checks client permissions keyed to the username in the client’s digital certificate. If the ACL requires a password, the client must provide the password when creating the connection object.

Server Parameters	To instruct clients to login using the usernames from their digital certificates, the EMS server must specify <code>true</code> for the <code>ssl_use_cert_username</code> parameter in <code>tibemsd.conf</code> . For more information, see SSL Server Parameters on page 137.
-------------------	--

Using a Special User Name to Log In

The EMS server can recognize one special username; when a client logs in with that username, the server authorizes the client with the username taken from the client's digital certificate.

Server Parameters	To configure this behavior, set the <code>ssl_cert_user_specname</code> parameter in <code>tibemsd.conf</code> . For more information, see SSL Server Parameters on page 137.
----------------------	---

Renegotiating the Session Key



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

When a server and a Java client establish an SSL connection, the client and the server agree on a symmetric key for encrypting and decrypting data they will exchange. In the default behavior, this key lasts for the lifetime of the session, but you can configure the server or client to renegotiate keys for client sessions (that is, replace them with new keys) based on elapsed time or on the amount of data exchanged.



Key renegotiation features apply only to Java client sessions. It is not available in other client APIs (such as .NET), nor in communications between two servers.



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

See Also

`ssl_renegotiate_size` on page 138

`ssl_renegotiate_interval` on page 138

`TibjmsContext.SSL_RENEGOTIATE_INTERVAL` on page 261

`TibjmsContext.SSL_RENEGOTIATE_SIZE` on page 261

File Names for Certificates and Keys

For all parameters that specify the identity (digital certificate), private key, issuer (certificate chain), or trusted list of certificate authorities, valid files must be specified. Not all types of files are supported for clients and servers. The description of each parameter details which formats it supports.

Table 39 lists the valid types of files.

Table 39 File types

Extension	Description
.pem	PEM encoded certificates and keys (allows the certificate and private key to be stored together in the same file)
.der	DER encoded certificates
.p8	PKCS#8 file
.p7b	PKCS#7 file
.p12	PKCS12 file (allows the certificate and private key to be stored together in the same file)
.jks	Java KeyStore file
.epf	Entrust store file

Configuring SSL in the Server

To use SSL, each instance of `tibemsd` must have a digital certificate and a private key. The server can optionally require a certificate chain or trusted certificates.

Set the server to listen for SSL connections from clients by using the `listen` parameter in `tibemsd.conf`. To specify that a port accept SSL connections, specify the SSL protocol in the `listen` parameter as follows:

```
listen = ssl://localhost:7243
```

SSL Parameters

Several SSL parameters can be set in `tibemsd.conf`. The minimum configuration is only one required parameter—`ssl_server_identity`. However, if the server's certificate file does not contain its private key, then you must specify it in `ssl_server_key`.

Within Table 18 on page 120, the section SSL Server Parameters on page 137 provides a complete description of the SSL parameters that can be set in `tibemsd.conf`.

Command Line Options

The server accepts a few command-line options for SSL.

When starting `tibemsd`, you can specify the following options:

- `-ssl_trace`—enables tracing of loaded certificates. This prints a message to the console during startup of the server that describes each loaded certificate.
- `-ssl_debug_trace`—enables more detailed SSL tracing for debugging only; it is not for use in production systems.
- `-ssl_password`—specifies the private key password. Alternatively, you can specify this password in the `ssl_server_password` parameter in `tibemsd.conf`. If you do not supply a password using either of these methods, `tibemsd` will prompt for the password when it starts. For more information, see the description of the `ssl_password` configuration parameter.

Configuring SSL in EMS Clients

To use an SSL connection to the EMS server, a Java client must include appropriate JAR files in the CLASSPATH (see Table 40 below). These files are included with EMS, and also with JDK (1.4 and later).

Table 40 SSL JAR Files

JAR File	Included with
jsse.jar	JDK
jnet.jar	JDK
jcrt.jar	JDK
tibcrypt.jar	EMS

To use Entrust with an EMS client, you must separately purchase and install the Entrust Version 7.1 libraries. If you use the Entrust libraries, you must include them in the CLASSPATH *before* the JSSE JAR files. To use Entrust Version 7.1 with JDK, you must download the unlimited strength policy JAR files from Sun's website and install them in your local installation of JDK. For installation and configuration details, see Entrust Version 7.1 documentation.

Client Digital Certificates

When client authentication with a digital certificate is required by the EMS server (see the description of the `ssl_require_client_cert` parameter in `tibemsd.conf`), the client may combine its client certificate and private key in a single file in one of the following formats:

- PKCS#12
- Java KeyStore
- Entrust Store

You can also store the private key file separately from the client certificate file. If this is the case, the certificate and private key must be stored in one of the following formats:

- PEM
- PKCS#8

The format of the client digital certificate and private key file depends on the SSL vendor used by the client. JSSE and Entrust support different formats and combinations of formats. For more information about formats, see your SSL vendor's documentation.

Configuring SSL

A Java client connecting to an EMS server can configure SSL characteristics in three ways:

- Use JNDI to lookup a connection factory object that specifies SSL connectivity. That is, the server URL in the connection factory must specify the SSL protocol, and the factory must specify appropriate SSL parameters.

A preconfigured connection factory is the preferred mechanism in many situations.

- Set global SSL parameters using the Java class `TibjmsSSL`.
- Create a connection factory object within program code. In C or COBOL use the type `tibemsSSLParams`.

Specifying any SSL parameters within a connection factory causes all global SSL parameters set with the `TibjmsSSL` class to be ignored.

Configuring a ConnectionFactory

You can configure a connection factory using the administration tool or the EMS Administration APIs. See Chapter 8, *Using the Administration Tool*, on page 161.

When configuring a connection factory, you can specify several SSL parameters. These parameters are similar to the server parameters that you can configure in `tibemsd.conf`.



When configuring a connection factory, EMS does not verify any file names specified in the SSL parameters. At the time the factory is retrieved using JNDI, the EMS server attempts to resolve any file references. If the files do not match the supported types or the files are not found, the JNDI lookup fails with a `ConfigurationException`.

Table 41 briefly describes the parameters you can set in a connection factory, and refers to additional information about each parameter. For more information about each parameter, see the description of the equivalent parameter in `tibemsd.conf`.

Table 41 *ConnectionFactory SSL parameters (Sheet 1 of 2)*

Parameter	Description
<code>ssl_vendor</code>	The vendor name of the SSL implementation that the client uses.
<code>ssl_identity</code>	<p>The client's digital certificate.</p> <p>For more information on file types for digital certificates, see File Names for Certificates and Keys on page 275.</p>
<code>ssl_issuer</code>	<p>Issuer's certificate chain for the client's certificate. Supply the entire chain, including the CA root certificate. The client reads the certificates in the chain in the order they are presented in this parameter.</p> <p>Example</p> <pre>ssl_issuer = certs\CA_root.pem ssl_issuer = certs\CA_child1.pem ssl_issuer = certs\CA_child2.pem</pre> <p>For more information on file types for digital certificates, see File Names for Certificates and Keys on page 275.</p>
<code>ssl_private_key</code>	<p>The client's private key. If the key is included in the digital certificate in <code>ssl_identity</code>, then you may omit this parameter.</p> <p>For more information on file types for digital certificates, see File Names for Certificates and Keys on page 275.</p>
<code>ssl_trusted</code>	<p>List of CA certificates to trust as issuers of server certificates. Supply only CA root certificates.</p> <p>For more information on file types for digital certificates, see File Names for Certificates and Keys on page 275.</p>
<code>ssl_verify_host</code>	<p>Specifies whether the client should verify the server's certificate. The values for this parameter are <code>enabled</code> or <code>disabled</code>. By default, this parameter is enabled, signifying the client should verify the server's certificate.</p> <p>When <code>disabled</code>, the client establishes secure communication with the server, but does not verify the server's identity.</p>

Table 41 *ConnectionFactory SSL parameters (Sheet 2 of 2)*

Parameter	Description
ssl_verify_hostname	<p>Specifies whether the client should verify the name in the CN field of the server’s certificate. The values for this parameter are <code>enabled</code> and <code>disabled</code>. By default, this parameter is <code>enabled</code>, signifying the client should verify the name of the connected host or the name specified in the <code>ssl_expected_hostname</code> parameter against the value in the server’s certificate. If the names do not match, the client rejects the connection.</p> <p>When <code>disabled</code>, the client establishes secure communication with the server, but does not verify the server’s name.</p>
ssl_expected_hostname	<p>The name the client expects in the CN field of the server’s certificate. If this parameter is not set, the expected name is the hostname of the server.</p> <p>The value of this parameter is used when the <code>ssl_verify_hostname</code> parameter is <code>enabled</code>.</p>
ssl_ciphers	<p>Specifies 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 more information, see Specifying Cipher Suites on page 281.</p>
ssl_rand_egd	<p>The path for the entropy gathering daemon (EGD), if one is installed. This daemon generates random data for the client.</p>

Specifying Cipher Suites

On the EMS server, specify cipher suites using the `ssl_server_ciphers` configuration parameter in `tibemsd.conf`. For more information about server configuration files, see Chapter 7, *Using the Configuration Files*, on page 117.

For clients connecting with a connection factory, specify cipher suites using the `ssl_ciphers` connection factory parameter. For more information, see *Configuring SSL in EMS Clients* on page 277.

Syntax for Cipher Suites

EMS uses OpenSSL for SSL support. Therefore, the cipher suite names can be specified as the OpenSSL name for the cipher suite.

When specifying cipher suites, the usual way to specify more than one cipher suite is to separate each suite name with a colon (:) character. Alternatively, you can use spaces and commas to separate names.

Java Client Syntax

The syntax for specifying the list of cipher suites is different for Java clients than for any other location where cipher suites can be specified. For Java clients, you specify a qualifier (for example, + to add the suite) followed by the cipher suite name. Cipher suite names are case-sensitive. Table 42 describes the qualifiers you can use when specifying cipher suite names in a `ConnectionFactory` for Java clients.

Table 42 Qualifiers for Cipher Suites in Java Clients

Qualifier	Description
+	Add the cipher to the list of ciphers.
-	Remove the cipher from the list of ciphers.
>	Move the cipher to the end of the list.
<	Move the cipher to the beginning of the list.
ALL	<p>All ciphers from the list (except null ciphers). You can use this keyword to add or remove all ciphers.</p> <p>At least one cipher suite must be present, otherwise the SSL connection fails to initialize. So, if you use -ALL, you must subsequently add the desired ciphers to the list.</p>

This example specifies cipher suites in the `ssl_ciphers` connection factory parameter in a Java client:

```
-ALL : +RC4-MD5 : +DES-CBC-SHA : <DES-CBC3-SHA
```

Syntax for All Other Cipher Suite Specifications

For any cipher suite list that is not specified in a connection factory of a Java client, use the OpenSSL syntax. In particular, C clients and the `ssl_server_ciphers` configuration parameter require OpenSSL syntax.

In OpenSSL syntax, specifying a cipher suite name adds that cipher suite to the list. Each cipher suite name can be preceded by a qualifier. Cipher suite names are case-sensitive. Table 43 describes the qualifiers available using OpenSSL syntax.

Table 43 *OpenSSL Qualifiers for Cipher Suites*

Qualifier	Description
/	Start with an empty list, and add the ciphers that follow it.
+	Moves the cipher to the end of the list.
-	Remove the cipher from the list of ciphers. When this option is used, the cipher can be added later on in the list of ciphers.
!	Permanently disable the cipher within the list of ciphers. Use this option if you wish to remove a cipher and you do not want later items in the list to add the cipher to the list. This qualifier takes precedence over all other qualifiers.
ALL	All ciphers from the list (except null ciphers). You can use this keyword to add or remove all ciphers. At least one cipher suite must be present or the SSL connection fails to initialize. So, after using <code>-ALL</code> , you should add at least one cipher to the list.
@STRENGTH	Sort the cipher list by key length.

This example specifies cipher suites in the `ssl_server_ciphers` configuration parameter.

```
ssl_server_ciphers = -ALL:RC4-MD5:DES-CBC-SHA:DES-CBC3-SHA
```

This example illustrates disables RC4-MD5, then adds all other ciphers:

```
ssl_server_ciphers = !RC4-MD5:ALL
```

Default Cipher List

The EMS server and C client library hard-code a default cipher list, which is equivalent to ALL: !ADH: RC4+RSA: +SSLv2: @STRENGTH.

Supported Cipher Suites

In general, the EMS server and C client library support all cipher suites that OpenSSL supports, except IDEA, RC-5 and CAST. For a complete list, see current OpenSSL documentation.

Java clients support *only* the cipher suites listed in Table 44. For convenience, the table lists both the standard name and the OpenSSL name for each cipher suite.

Table 44 Supported Cipher Suites in Java API (Sheet 1 of 3)

Suite Name (OpenSSL Name)	Export	Key Exch	Auth	Encrypt	Key Size	MAC
SSL_RSA_WITH_RC4_128_MD5 (RC4-MD5)		RSA	RSA	RC4	128	MD5
SSL_RSA_WITH_RC4_128_SHA (RC4-SHA)		RSA	RSA	RC4	128	SHA1
SSL_RSA_WITH_DES_CBC_SHA (DES-CBC-SHA)		RSA	RSA	DES	56	SHA1
SSL_RSA_WITH_3DES_EDE_CBC_SHA (DES-CBC3-SHA)		RSA	RSA	3-DES	168	SHA1
SSL_RSA_EXPORT_WITH_RC4_40_MD5 (EXP-RC4-MD5)	Yes	RSA(512)	RSA	RC4	40	MD5
SSL_RSA_EXPORT_WITH_DES_40_CBC_SHA (EXP-DES-CBC-SHA)	Yes	RSA(512)	RSA	DES	40	SHA1

Table 44 Supported Cipher Suites in Java API (Sheet 2 of 3)

Suite Name (OpenSSL Name)	Export	Key Exch	Auth	Encrypt	Key Size	MAC
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA (EDH-DSS-DES-CBC3-SHA)						
		DH	DSS	3-DES	168	SHA1
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA (EDH-RSA-DES-CBC3-SHA)						
		DH	RSA	3-DES	168	SHA1
SSL_DHE_DSS_WITH_DES_CBC_SHA (EDH-DSS-DES-CBC-SHA)						
		DH	DSS	DES	56	SHA1
SSL_DHE_RSA_WITH_DES_CBC_SHA (EDH-RSA-DES-CBC-SHA)						
		DH	RSA	DES	56	SHA1
SSL_DHE_DSS_EXPORT_WITH_DES_40_CBC_SHA (EXP-EDH-DSS-DES-CBC-SHA)						
	Yes	DH(512)	DSS	DES	40	SHA1
SSL_DHE_RSA_EXPORT_WITH_DES_40_CBC_SHA (EXP-EDH-RSA-DES-CBC-SHA)						
	Yes	DH(512)	RSA	DES	40	SHA1
TLS_RSA_WITH_AES_128_CBC_SHA (AES128-SHA)						
		RSA	RSA	AES	128	SHA1
TLS_RSA_WITH_AES_256_CBC_SHA (AES256-SHA)						
		RSA	RSA	AES	256	SHA1

Table 44 Supported Cipher Suites in Java API (Sheet 3 of 3)

Suite Name (OpenSSL Name)	Export	Key Exch	Auth	Encrypt	Key Size	MAC
TLS_DHE_DSS_WITH_AES_128_CBC_SHA (DHE-DSS-AES128-SHA)		DH	DSS	AES	128	SHA1
TLS_DHE_DSS_WITH_AES_256_CBC_SHA (DHE-DSS-AES256-SHA)		DH	DSS	AES	256	SHA1
TLS_DHE_RSA_WITH_AES_128_CBC_SHA (DHE-RSA-AES128-SHA)		DH	RSA	AES	128	SHA1
TLS_DHE_RSA_WITH_AES_256_CBC_SHA (DHE-RSA-AES256-SHA)		DH	RSA	AES	256	SHA1
SSL_RSA_WITH_NULL_MD5 (NULL-MD5)						
JSSE only. Entrust does not support this suite.						
		DH	RSA	none	—	MD5
SSL_RSA_WITH_NULL_SHA (NULL-SHA)						
JSSE only. Entrust does not support this suite.						
		DH	RSA	none	—	SHA1

SSL Authentication Only

EMS servers can use SSL for secure data exchange (standard usage), or only for client authentication. This section describes the use of SSL for client authentication.

Motivation Some applications require strong or encrypted authentication, but do not require message encryption.

In this situation, application architects could configure SSL with a null cipher. However, this solution incurs internal overhead costs of SSL calls, decreasing message speed and throughput.

For optimal performance, the preferred solution is to use SSL only to authenticate clients, and then avoid SSL calls thereafter, using ordinary TCP communications for subsequent data exchange. Message performance remains unaffected.

Preconditions All three of these preconditions must be satisfied to use SSL only for authentication:

- The server and clients must both be release 4.2 or later. (If not, EMS behavior reverts to using SSL for all communications throughout the life of the connection.)
- The server must explicitly enable the parameter `ssl_auth_only` in the main configuration file.
- The client program must request a connection that uses SSL for authentication only. Administrators can specify this request in factories by enabling the parameter `ssl_auth_only`, or programs can call the Java method `TibemsSSL.setAuthOnly` (or the equivalent C function, `tibemsSSLParams_SetAuthOnly`).

See Also `ssl_auth_only` on page 142

Third-Party SSL Hardware Accelerators



Support for SSL accelerators was deprecated as of release 4.1.0. In a future release, this feature will become obsolete and unsupported.

While the SSL protocol provides message security and integrity, the connection handshake and bulk message encryption can require significant machine resources. To reduce this overhead, you can deploy a third-party SSL hardware accelerator. The TIBCO Enterprise Message Service server supports external (rack-mount) hardware accelerators. SSL accelerators are capable of off-loading the main CPU from asymmetric public/private key negotiations as well as well as secret key bulk message encryption and decryption.

Ingrian Accelerator

Ingrian provides a variety of accelerator products, such as the Ingrian i100, i140, i210, and so on. These products are external hardware accelerators that fit into standard rack mount space. The Ingrian Accelerator is placed on the network between the client and the server and off loads all SSL functionality from the server. The clients use SSL to communicate with the server by connecting to an SSL port on the Ingrian Accelerator. The Ingrian Accelerator completely performs the SSL handshake and passes messages to the server over TCP. Figure 17 illustrates the operation of the Ingrian Accelerator.

Figure 17 Ingrian Accelerator



Because the Ingrian Accelerator performs all SSL functionality, it must be separately configured for listen ports, certificates, and so on. See www.ingrian.com for more information about Ingrian Accelerator products, and refer to the documentation for the specific accelerator for information about how to configure the accelerator for your application.

Configuring an Ingrian Accelerator to Communicate with a TIBCO EMS Server

The Ingrian Accelerator is configured to pass data from a client to a server through *forwarding rules*. A forwarding rule specifies, among other things, the listen port that clients connect to, the target port on the back-end server, the protocol that clients use to communicate with the Ingrian Accelerator, and the protocol that the accelerator uses to communicate with the back end server.

The Ingrian Accelerator supports several protocols. To be used with the TIBCO Enterprise Message Service server however, the accelerator must be configured to use the “SSL Any” protocol to communicate with clients and the “Any” protocol to communicate with the server. See the documentation for the specific Ingrian Accelerator you are using for information about how to configure forwarding rules.

The Ingrian Accelerator can also be configured to extract the user name from the client certificate and pass it to the server for user authentication. If you require this functionality, please contact TIBCO for instructions about how to enable this feature in the Ingrian Accelerator and the TIBCO Enterprise Message Service server.

Because the Ingrian Accelerator off loads the server of all SSL functionality, none of the parameters in `tibemsd.conf` for configuring SSL in the server are used. When the Ingrian Accelerator is deployed, the TIBCO Enterprise Message Service server should be configured to use standard TCP communication.

Chapter 13 **Fault Tolerance**

This chapter describes the fault tolerance features of TIBCO Enterprise Message Service.

Topics

- *Fault Tolerance Overview, page 290*
- *Failover, page 292*
- *Shared State, page 295*
- *Configuring Fault-Tolerant Servers, page 299*
- *Configuring Clients for Fault-Tolerant Connections, page 301*

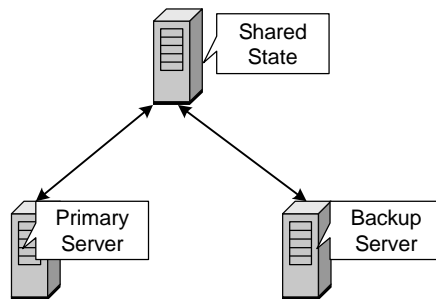
Fault Tolerance Overview

You can arrange TIBCO Enterprise Message Service servers for fault-tolerant operation by configuring a pair of servers—one primary and one backup. The primary server accepts client connections, and interacts with clients to deliver messages. If the primary server fails, the backup server resumes operation in its place. (We do not support more than two servers in a fault-tolerant configuration.)

Shared State A pair of fault-tolerant servers must have access to shared state, which consists of information about client connections and persistent messages. This information enables the backup server to properly assume responsibility for those connections and messages.

Figure 18 illustrates a fault-tolerant configuration of TIBCO Enterprise Message Service.

Figure 18 Primary Server and Backup Server



Locking To prevent the backup server from assuming the role of the primary server, the primary server locks the shared state during normal operation. If the primary server fails, the lock is released, and the backup server can obtain the lock.

Configuration Files

When a primary server fails, its backup server assumes the status of the primary server and resumes operation. Before becoming the new primary server, the backup server re-reads all of its configuration files. If the two servers share configuration files, then administrative changes to the old primary carry over to the new primary.



When fault-tolerant servers share configuration files, you *must* limit configuration changes to the current primary server only. Separately reconfiguring the backup server can cause it to overwrite the shared configuration files; unintended misconfiguration can result.

Failover

This section presents details of the failover sequence.

Detection

A backup server detects a failure of the primary in either of two ways:

- *Heartbeat Failure*—The primary server sends heartbeat messages to the backup server to indicate that it is still operating. When a network failure stops the servers from communicating with each other, the backup server detects the interruption in the steady stream of heartbeats. For details, see *Heartbeat Parameters* on page 294.
- *Connection Failure*—The backup server can detect the failure of its TCP connection with the primary server. When the primary process terminates unexpectedly, the backup server detects the broken connection.

Response

When a backup server (B) detects the failure of the primary server (A), then B attempts to assume the role of primary server. First, B obtains the lock on the current shared state. When B can access this information, it becomes the new primary server.

Role Reversal	When B becomes the new primary server, A can restart as a backup server, so that the two servers exchange roles.
Client Transfer	Clients of A that are configured to failover to backup server B automatically transfer to B when it becomes the new primary server. B reads the client's current state from the shared storage to deliver any persistent messages to the client.
Client Notification	Client applications can receive notification when failover occurs.



To receive notification, Java client programs can set the system property `tibco.tibjms.ft.switch.exception` to any value, and define an `ExceptionListener` to handle failover notification; see the class `com.tibco.tibjms.Tibjms` in *TIBCO Enterprise Message Service Java API Reference*.

To receive notification, .NET client programs can call `Tibems.SetExceptionOnFTSwitch(true)`, and define an exception listener to handle failover notification; see the method `Tibems.SetExceptionOnFTSwitch` on page 220 in *TIBCO Enterprise Message Service .NET API Reference*.

Lock Unavailable If B cannot obtain the lock immediately, it alternates between attempting to obtain the lock (and become the primary server), and attempting to reconnect to A (and resume as a backup server)—until one of these attempts succeeds.

Message Redelivery

Persistent	When a failure occurs, messages with delivery mode <code>PERSISTENT</code> , that were not successfully acknowledged before the failure, are redelivered.
Failsafe	EMS guarantees that a message with <code>PERSISTENT</code> delivery mode and a failsafe destination will not be lost during a failure.
Delivery Succeeded	Any messages that have been successfully acknowledged or committed are not redelivered, in compliance with the JMS 1.1 specification.
Topics	All topic subscribers continue normal operation after a failover.

Queues

For queue receivers, any messages that have been sent to receivers, but have not been acknowledged before the failover, may be sent to other receivers immediately after the failover.

A receiver trying to acknowledge a message after a failover may receive the `javax.jms.IllegalStateException`. This exception signifies that the attempted acknowledgement is for a message that has already been sent to another queue receiver. This exception only occurs in this scenario, or when the session or connection have been closed. This exception cannot occur if there is only one receiver at the time of a failover, but it may occur for exclusive queues if more than one receiver was started for that queue.

When a queue receiver catches a `javax.jms.IllegalStateException`, the best course of action is to call the `Session.recover()` method. Your application program should also be prepared to handle redelivery of messages in this situation. All queue messages that can be redelivered to another queue receiver after a failover always have the header field `JMSRedelivered` set to `true`; application programs must check this header to avoid duplicate processing of the same message in the case of redelivery.



Acknowledged messages are never redelivered (in compliance with the JMS specification). The case described above occurs when the application cannot acknowledge a message because of a failover.

Transactions

A transaction is considered *active* when at least one message has been sent or received by the session, and the transaction has not been successfully committed.

After a failover, attempting to commit the active transaction results in a `javax.jms.TransactionRolledBackException`. Clients that use transactions must handle this exception, and resend any messages sent during the transaction. The backup server automatically redelivers any messages that were delivered to the session during the transaction that rolled back.

Heartbeat Parameters

When the primary server heartbeat stops, the backup server waits for its activation interval (elapsed time since it detected the most recent heartbeat); then the backup server retrieves information from shared storage and assumes the role of primary server.

The default heartbeat interval is 3 seconds, and the default activation interval is 10 seconds. The activation interval must be at least twice the heartbeat interval. Both intervals are specified in seconds. You can set these intervals using the administration tool or in the server configuration files.

For more information about these parameters, see Fault Tolerance Parameters on page 127.

Shared State

The primary server and backup server must share the same state. Server state includes three categories of information:

- persistent message data (for queues and topics)
- client connections of the primary server
- metadata about message delivery

During a failover, the backup server re-reads all shared state information.

Implementing Shared State

We recommend that you implement shared state using shared storage devices. The shared state must be accessible to both the primary and backup servers.

Support Criteria

Several options are available for implementing shared storage using a combination of hardware and software. EMS requires that your storage solution guarantees all four criteria in Table 45.



Always consult your shared storage vendor *and* your operating system vendor to ascertain that the storage solution you select satisfies all four criteria.

Table 45 Shared Storage Criteria for Fault Tolerance (Sheet 1 of 2)

Criterion	Description
Write Order	The storage solution must write data blocks to shared storage in the same order as they occur in the data buffer. (Solutions that write data blocks in any other order (for example, to enhance disk efficiency) do <i>not</i> satisfy this requirement.)
Synchronous Write Persistence	Upon return from a synchronous write call, the storage solution guarantees that all the data have been written to durable, persistent storage.

Table 45 Shared Storage Criteria for Fault Tolerance (Sheet 2 of 2)

Criterion	Description
Distributed File Locking	<p>The EMS servers must be able to request and obtain an exclusive lock on the shared storage. The storage solution must <i>not</i> assign the locks to two servers simultaneously. (See Software Options on page 297.)</p> <p>EMS servers use this lock to determine the primary server.</p>
Unique Write Ownership	<p>The EMS server process that has the file lock must be the only server process that can write to the file. Once the system transfers the lock to another server, pending writes queued by the previous owner must fail.</p>

Hardware Options

Consider these examples of commonly-sold hardware options for shared storage:

- Dual-Port SCSI device
- Storage Area Network (SAN)
- Network Attached Storage (NAS)

SCSI and SAN	Dual-port SCSI and SAN solutions generally satisfy the Write Order and Synchronous Write Persistence criteria. (The clustering software must satisfy the remaining two criteria.) As always, you must confirm all four requirements with your vendors.
NAS	<p>NAS solutions require a CS (rather than a CFS) to satisfy the Distributed File Locking criterion (see below).</p> <p>Some NAS solutions satisfy the criteria, and some do not; you must confirm all four requirements with your vendors.</p>
NAS with NFS	<p>When NAS hardware uses NFS as its file system, it is particularly difficult to determine whether the solution meets the criteria. Our research indicates the following conclusions:</p> <ul style="list-style-type: none">• NFS v2 definitely does <i>not</i> satisfy the criteria.• NFS v3 with UDP definitely does <i>not</i> satisfy the criteria.• NFS v3 with TCP <i>might</i> satisfy the criteria. Consult with the NAS vendor to verify that the NFS server (in the NAS) satisfies the criteria. Consult with the operating system vendor to verify that the NFS client (in the OS on the server host computer) satisfies the criteria. When both vendors certify that their

components cooperate to guarantee the criteria, then the shared storage solution supports EMS.

Software Options

Consider these examples of commonly-sold software options:

- Cluster Server (CS)
A cluster server monitors the EMS server processes and their host computers, and ensures that exactly one server process is running at all times. If the primary server fails, the CS restarts it; if it fails to restart the primary, it starts the backup server instead.
- Clustered File System (CFS)
A clustered file system lets the two EMS server processes run simultaneously. It even lets both servers mount the shared file system simultaneously. However, the CFS assigns the lock to only one server process at a time. The CFS also manages operating system caching of file data, so the backup server has an up-to-date view of the file system (instead of a stale cache).

With dual-port SCSI or SAN hardware, either a CS or a CFS might satisfy the Distributed File Locking criterion. With NAS hardware, only a CS can satisfy this criterion (CFS software generally does not). Of course, you must confirm all four requirements with your vendors.

Messages Stored in Shared State

Messages with PERSISTENT delivery mode are stored, and are available in the event of primary server failure. Messages with NON_PERSISTENT delivery mode are not available if the primary server fails.

For more information about recovery of messages during failover, see Message Redelivery on page 293.

Storage Files

The `tibemsd` server creates three files to store shared state.

Table 46 Shared State Files

File Name	Description
<code>meta.db</code>	This file records durable subscribers, fault-tolerant connections, and other metadata.

Table 46 Shared State Files

File Name	Description
sync-msgs.db	When a queue or topic definition (in a configuration file) specifies that the destination is <code>failsafe</code> , then the server stores its messages in this file (using synchronous I/O calls).
async-msgs.db	When a queue or topic definition (in a configuration file) does <i>not</i> specify that the destination is <code>failsafe</code> , then the server stores its messages in this file (using asynchronous I/O calls).

For more information about the `failsafe` destination property, see `failsafe` on page 35.

For more information about configuration files, see Chapter 7, Using the Configuration Files, on page 117.33

Storage Parameters

Several configuration parameters apply to EMS storage files (even when fault-tolerant operation is not configured); see Storage Files on page 121.

Configuring Fault-Tolerant Servers

To configure an EMS server as a fault-tolerant backup, set these parameters in its main configuration file (or on the server command line):

- `server` Set this parameter to the same server name in the configuration files of both the primary server and the backup server.
- `ft_active` In the configuration file of the primary server, set this parameter to the URL of the backup server. In the configuration file of the backup server, set this parameter to the URL of the primary server.

When the backup server starts, it attempts to connect to the primary server. If it establishes a connection to the primary, then the backup server enters standby mode. If it cannot establish a connection to the primary, then the backup server assumes the role of the primary server (in active mode).

While the backup server is in standby mode, it does not accept connections from clients. To administer the backup server, the `admin` user can connect to it using the administration tool.

Authorization and Fault-Tolerant Servers

EMS authorization interacts with fault tolerance. If `authorization` is enabled, then both servers in a fault-tolerant pair must have the same server name (that is, the `server` parameter in `tibemsd.conf`). Furthermore, you must add that server name as a user in the configuration file `users.conf`.

SSL

You can use SSL to secure communication between a pair of fault-tolerant servers.

Parameters in the main configuration file (`tibemsd.conf`) affect this behavior; see Fault Tolerance Parameters on page 127. The relevant parameters all begin with the prefix `ft_ssl`. You must configure these parameters for both servers in the pair.

See Also Chapter 12, Using the SSL Protocol, on page 265

Reconnect Timeout

When a backup server assumes the role of the primary server during failover, clients attempt to reconnect to the backup server (that is, the new primary) and continue processing their current message state. As each client reconnects, the backup server reads its message state from the shared state files.

You can instruct the server to clean up state information for clients that do not reconnect before a specified time limit.

The `ft_reconnect_timeout` configuration parameter specifies that time limit (in seconds). The default value is 60 seconds. See also, Fault Tolerance Parameters on page 127.

Configuring Clients for Fault-Tolerant Connections

When a backup server assumes the role of the primary server during failover, clients attempt to reconnect to the backup server (that is, the new primary). To enable a client to reconnect, you must specify the URLs of both servers. Specify multiple servers as a comma-separated list of URLs. Both URLs must use the same protocol (either `tcp` or `ssl`).



The client attempts to connect to each URL in the order listed. If a connection to one URL fails, the client tries the next URL in the list. The client tries the URLs in sequence until all URLs have been tried; if the first failed connection was not the first URL in the list, the attempts wrap to the start of the list (so each URL is tried). If none of the attempts succeed, the connection fails.

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

Using TibjmsConnectionFactory in Java

Java clients can list the primary and backup server URLs in the `serverURL` argument to a `TibjmsConnectionFactory` constructor. For example:

```
TibjmsTopicConnectionFactory factory = new
TibjmsTopicConnectionFactory(
    "tcp://server0:7222, tcp://server1:7344");
```

ConnectionFactoryInfo Constructor in Java

Java clients can use the Java Administrator API to specify the primary and backup server URLs in the `url` argument to a `ConnectionFactoryInfo` constructor. For example:

```
ConnectionFactoryInfo cfi = new
ConnectionFactoryInfo(
    "ssl://server0:7222,
    ssl://server1:7344,
    ssl://server2:7222",
    null, QUEUE_TYPE, params );
```

Connection Constructor in C

C clients list the primary and backup server URLs in the `brokerURL` argument to a connection constructor. For example:

```
tibjmsQueueConnection_Create(
    &qc,
```

```
"tcp://server0:7222,  
tcp://server1:7344",  
NULL, "admin", NULL );
```

JNDI Look-Up

When creating a connection factory using `tibemsadmin`, an administrator can specify multiple server URLs in the `url` argument of the `create factory` command. For example:

```
create factory myFactory topic  
url=tcp://server0:7545,tcp://server1:7344,tcp://server2:7433
```

Specifying More Than Two URLs

Even though there are only two servers (the primary and backup servers), clients can specify more than two URLs for the connection. For example, if each server has more than one listen address, a client can reconnect to the same server at a different address (that is, at a different network interface). In another example, clients can configure to failover to parallel remote servers for catastrophe recovery.

Chapter 14 Working With Routes

This chapter describes routing of messages among TIBCO Enterprise Message Service servers.

Topics

- *Overview of Routing, page 304*
- *Zone, page 308*
- *Active and Passive Routes, page 311*
- *Configuring Routes and Zones, page 312*
- *Routed Topic Messages, page 317*
- *Routed Queues, page 322*
- *Routing and Authorization, page 325*

Overview of Routing

TIBCO Enterprise Message Service servers can route messages to other servers.

- Topic messages can travel one hop or multiple hops (from the first server).
- Queue messages can travel only one hop *to* the home queue, and one hop *from* the home queue.

You can define routes using an administrative interface (that is, configuration files, `tibemsadmin`, or administration APIs).

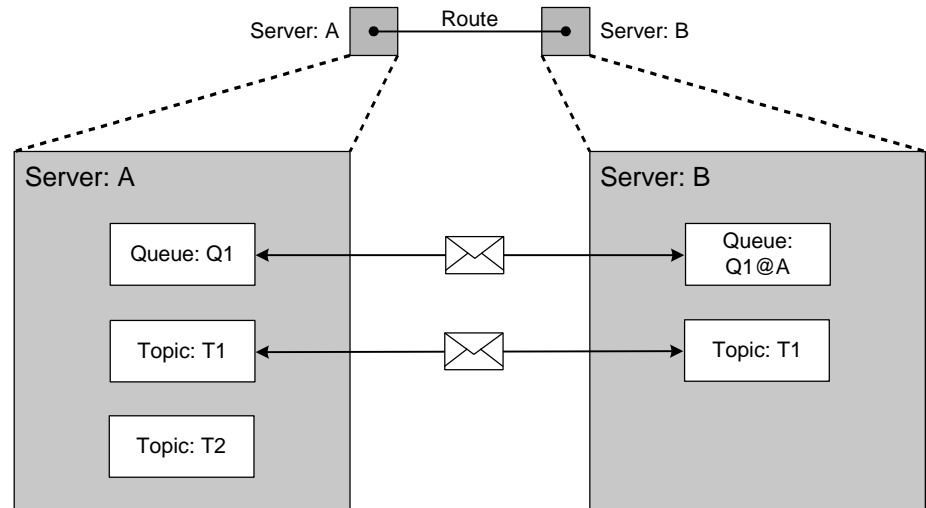
Route

Basic Operation

- Each *route* connects two TIBCO Enterprise Message Service servers.
- Each route forwards messages between corresponding destinations (that is, global topics with the same name, or explicitly routed queues) on its two servers.
- Routes are bidirectional; that is, each server in the pair forwards messages along the route to the other server.

For example, the compact view at the top of Figure 19 denotes a route between two servers, A and B. The exploded view beneath it illustrates the behavior of the route. Each server has a global topic named T1, and a routed queue Q1; these destinations correspond, so the route forwards messages between them. In addition, server A has a global topic T2, which does not correspond to any topic on server B. The route does not forward messages from T2.

Figure 19 Routes: bidirectionality and corresponding destinations

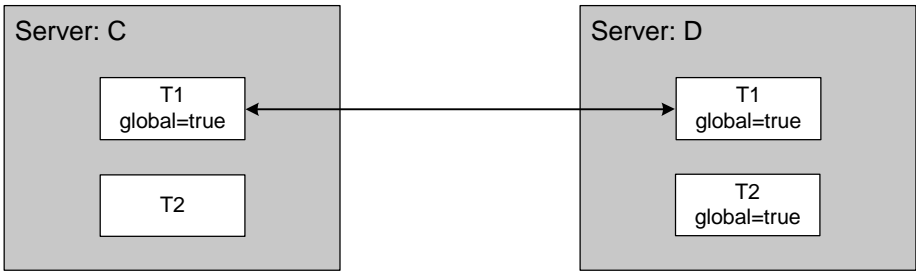


Global Destinations

Routes forward messages only between global destinations—that is, for topics the `global` property must be true on both servers (for queues, see Routed Queues on page 322). This rule overrides the inherent bidirectionality of routes. (For more information about destination properties, See Destination Properties on page 34.)

Figure 20 illustrates a route between two servers, C and D, with corresponding destinations T1 and T2. Notice that T1 is global on both C and D, so both servers forward messages across the route to the corresponding destination. However, T2 is not global on C, neither C nor D forward T2 messages to one another.

Figure 20 Routes: global destinations



Unique Routing Path

It is illegal to define a set of routes that permit a message to reach a server by more than one path. TIBCO Enterprise Message Service servers detect illegal duplicate routes and report them as configuration errors.

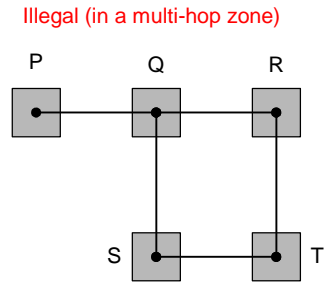
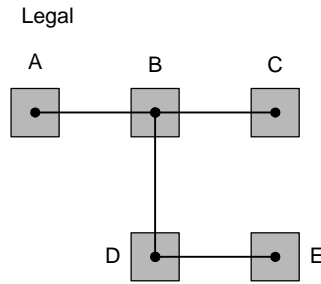
Figure 21 on page 307 depicts two sets of routes. On the left, the routes connecting servers A, B, C, D and E form an acyclic graph, with only one route connecting any pair of servers; this configuration is legal (in any zone).

In contrast, the routing configuration on the right is illegal in a multi-hop zone. The graph contains redundant routing paths between servers Q and S (one direct, and one through R and T).



Note that the configuration on the right is illegal only in a multi-hop zone; it is legal in a one-hop zone. For further information, see Zone on page 308.

Figure 21 Routes: Unique Path



Zone

Zones restrict the behavior of routes, so you can configure complex routing paths. Zones affect topic messages, but not queue messages.

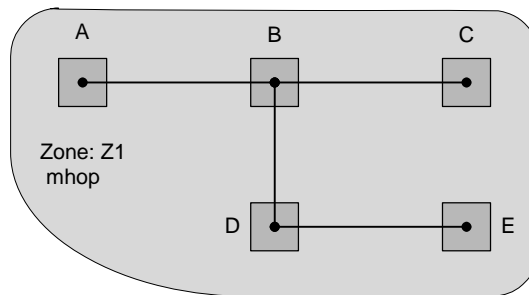
Basic Operation

A *zone* is a named set of routes. Every route belongs to a zone. A zone affects the forwarding behavior of its routes:

- In a multi-hop (`mhop`) zone, topic messages travel along all applicable routes to all servers connected by routes within the zone.
- In a one-hop (`1hop`) zone, topic messages travel only one hop (from the first server).
- Queue messages travel only one hop, even within multi-hop zones.

For example, Figure 22 depicts a set of servers connected by routes within a multi-hop zone, Z1. If a client sends a message to a global topic on server B, the servers forward the message to A, C, D and E. In contrast, if Z1 were a one-hop zone, B would forward the message to A, C and D—but D would *not* forward it E.

Figure 22 Zones: multi-hop



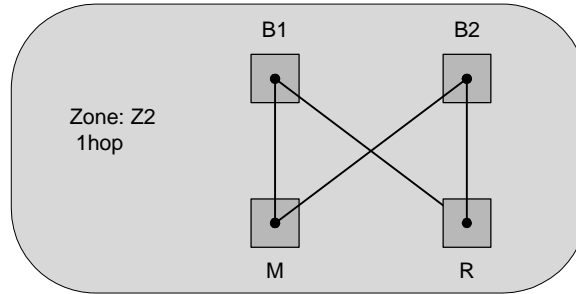
Eliminating Redundant Paths with a One-Hop Zone

Figure 23 illustrates an enterprise with four servers:

- B1 and B2 serve producers at branch offices of an enterprise.
- M serves consumers at the main office, which process the messages from the branches.
- R serves consumers that record messages for archiving, auditing, and backup.

The goal is to forward messages from B1 and B2 to both M and R. The routing graph *seems* to contain a cycle—the path from B1 to M to B2 to R duplicates the route from B1 to R. However, since these routes belong to the one-hop zone Z2, it is impossible for messages to travel the longer path. Instead, this limitation results in the desired result—forwarding from B1 to M and R, and from B2 to M and R.

Figure 23 Zones: one-hop



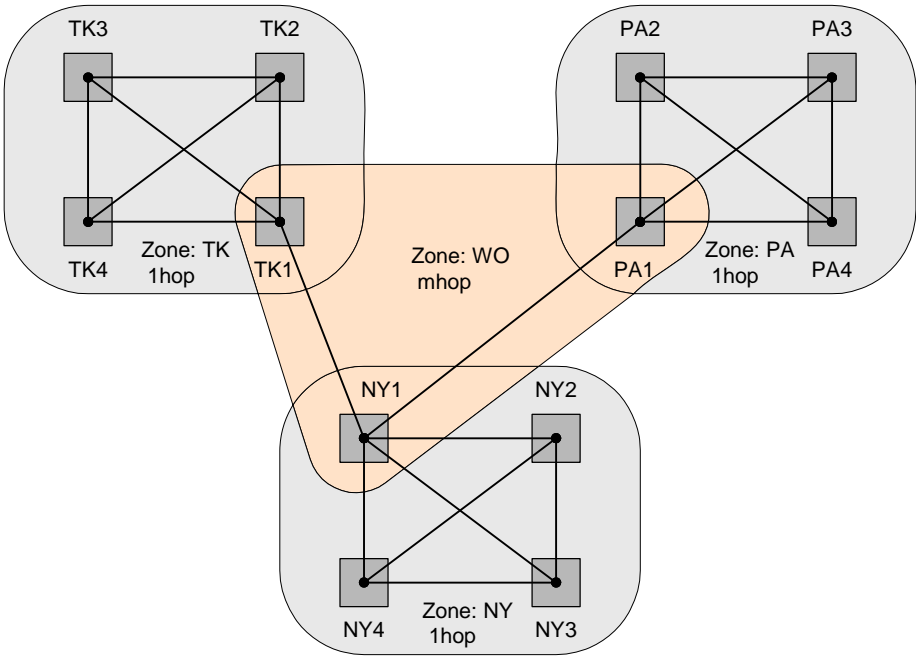
Overlapping Zones

A server can have routes that belong to several zones. When zones overlap at a server, the routing behavior within each zone does not limit routing in other zones. That is, when a forwarded message reaches a server with routes in several zones, the message crosses zone boundaries, and its hop count is reset to zero.

Figure 24 on page 310 illustrates an enterprise with one-hop zones connecting all the servers in each of several cities in a fully-connected graph. Zone TK connects all the servers in Tokyo; zone NY connects all the servers in New York; zone PA connects all the servers in Paris. In addition, the multi-hop zone WO connects one server in each city.

When a client of server TK3 produces a message, it travels one hop to each of the other Tokyo servers. When the message reaches TK1, it crosses into zone WO. TK1 forwards the message to NY1, which in turn forwards it to PA1. When the message reaches NY1, it crosses into zone NY (with hop count reset to zero); NY1 forwards it one hop to each of the other New York servers. Similarly, when the message reaches PA1, it crosses into zone PA (with hop count reset to zero); PA1 forwards it one hop to each of the other Paris servers.

Figure 24 Zones: overlap



Active and Passive Routes

A route connects two servers. You may configure a route at either or both of the servers.

Active-Passive Routes

When you configure a route at only one server, this asymmetry results in two perspectives on the route:

- A route is *active* from the perspective of the server where it is configured. This server actively initiates the connection to the other server, so we refer to it as the *active server*, or *initiating server*.
- A route is *passive* from the perspective of the other server. This server passively accepts connection requests from the active server, so we refer to it as the *passive server*.

A server can have both active and passive routes. That is, you can configure server S to initiate routes, and also configure other servers to initiate routes to S.

You can specify and modify the properties of an active route, but not those of a passive route. That is, properties of routes are associated with the server where the route is configured, and which initiates the connection.



Note that defining a route specifies a zone as well (either implicitly or explicitly). The first route in the zone defines the type of the route; subsequent routes in the same zone must have the same zone type (otherwise, the server reports an error).

Active-Active Routes

Two servers can both configure an active route one to the other. This arrangement is called an *active-active configuration*. For example, server A specifies a route to server B, and B specifies a route to A. Either server can attempt to initiate the connection. This configuration results in only one connection; it does *not* result in redundant routes.

You can promote an *active-passive* route to an *active-active* route. To promote a route, use this command on the passive server:

```
create route name url=url
```

The *url* argument is required, so that the server (where the route is being promoted) can connect to the other server if the route becomes disconnected. See also `create route` on page 170.

The promoted route behaves as a statically configured route—that is, it persists messages for durable subscribers, and stores its configuration in `routes.conf`, and administrators can modify its properties.

Configuring Routes and Zones

You can create routes using the administration tool (see Chapter 8 on page 161), or the administration APIs (see `com.tibco.tibjms.admin.RouteInfo` in the online documentation).

Syntax To create a route using the administration tool, first connect to one of the servers, then use the `create route` command with the following syntax:

```
create route <name> url=<URL> zone_name=<zone_name> zone_type=1hop|mhops <properties>
```

- `<name>` is the name of the passive server (at the other end of the route); it also becomes the name of the route.
- `<URL>` specifies the passive server by its URL—including protocol and port.

If your environment is configured for fault tolerance, the URL can be a comma-separated list of URLs denoting fault-tolerant servers. For more information about fault tolerance, see Chapter 13, Fault Tolerance, on page 289.

- `<zone_name>` specifies that the route belongs to the routing zone with this name. When absent, the default value is `default_mhops_zone` (this default yields backward compatibility with configurations from releases earlier than 4.0).
- The zone type is either `1hop` or `mhops`. When omitted, the default value is `mhops`.
- `<properties>` is a space-separated list of properties for the route. Each property has the syntax `<prop_name>=<value>`.

For gating properties that control the flow of topics along the route, see Selectors for Routing Topic Messages on page 319.

For properties that configure the Secure Sockets Layer (SSL) protocol for the route, see Routing and SSL on page 313.

Example For example, these commands on server A would create routes to servers B and C. The route to B belongs to the one-hop zone Z1. The route to C belongs to the multi-hop zone ZM.

```
create route B url=tcp://B:7454 zone_name=Z1 zone_type=1hop
create route C url=tcp://C:7455 zone_name=ZM zone_type=mhops
```

Show Routes You can display these routes using the `show routes` command in the administration tool:

show routes					
Route	T	ConnID	URL	Zone	T
B	A	3	tcp://B:7454	Z1	1

C	A	-	tcp://C:7455	ZM	m
---	---	---	--------------	----	---

- The **Route** column lists the name of the passive server.
- The **T** column indicates whether the route is active (A) or passive (P), from the perspective of server A.
- The **ConnID** column contains either an integer connection ID (if the route is currently connected, or a dash (-) if the route is not connected.

Routes to Fault-Tolerant Servers

You can configure servers for fault tolerance. Client applications can specify the primary and backup servers; if the client's connection to the primary server fails, the client can connect to the backup server and resume operation. Similarly, a route specification can specify primary and secondary passive servers, so that if the route to the primary server fails, the active server can connect to the backup server and resume routing.

Failover behavior for route connections is similar to that for client connections; see *Configuring Clients for Fault-Tolerant Connections* on page 301.

Example

```
create route B url=tcp://B:7454,tcp://BBackup:7454 zone_name=Z1 zone_type=1hop
```

Routing and SSL

When configuring a route, you can specify SSL parameters for the connection. Although both participants in an SSL connection must specify a similar set of parameters, each server specifies this information in a different place:

- The passive server must specify SSL parameters in its main configuration file, `tibemsd.conf`.
- When a server initiates an SSL connection, it sends the route's SSL parameters to identify and authenticate itself to the passive server. You can specify these parameters when creating the route, or you can specify them in the route configuration file, `routes.conf`.

Table 47 lists the parameters that you can specify in the `routes.conf` configuration file, or on the command line when creating a route. The parameters for configuring SSL between routed servers are similar to the parameters used to configure SSL between server and clients; see Chapter 12, *Using the SSL Protocol*, on page 265.

Table 47 SSL Parameters for Routes (Sheet 1 of 3)

Parameter	Description
<code>ssl_identity</code>	<p>The server’s digital certificate in PEM, DER, or PKCS#12 format. You can copy the digital certificate into the specification for this parameter, or you can specify the path to a file that contains the certificate in one of the supported formats.</p> <p>For more information, see <i>File Names for Certificates and Keys</i> on page 275.</p>
<code>ssl_issuer</code>	<p>Certificate chain member for the server. Supply the entire chain, including the CA root certificate. The server reads the certificates in the chain in the order they are presented in this parameter.</p> <p>The certificates must be in PEM, DER, PKCS#7 or PKCS#12 format.</p> <p>Example</p> <pre>ssl_issuer = certs\CA_root.pem ssl_issuer = certs\CA_child1.pem ssl_issuer = certs\CA_child2.pem</pre> <p>For more information, see <i>File Names for Certificates and Keys</i> on page 275.</p>
<code>ssl_private_key</code>	<p>The local server’s private key. If the digital certificate in <code>ssl_identity</code> already includes this information, then you may omit this parameter.</p> <p>This parameter accepts private keys in PEM, DER and PKCS#12 formats.</p> <p>You can specify the actual key in this parameter, or you can specify a path to a file that contains the key.</p> <p>For more information, see <i>File Names for Certificates and Keys</i> on page 275.</p>

Table 47 SSL Parameters for Routes (Sheet 2 of 3)

Parameter	Description
ssl_password	<p>Private key or password for private keys.</p> <p>You can set passwords using the <code>tibemsadmin</code> tool. When passwords are set with this tool, the password is obfuscated in the configuration file. For more information, see Chapter 8, Using the Administration Tool, on page 161.</p>
ssl_trusted	<p>List of certificates that identify trusted certificate authorities.</p> <p>The certificates must be in PEM, DER or PKCS#7 format. You can either provide the actual certificates, or you can specify a path to a file containing the certificate chain.</p> <p>For more information, see File Names for Certificates and Keys on page 275.</p>
ssl_verify_host	<p>Specifies whether the server must verify the other server's certificate. The values for this parameter are <code>enabled</code> and <code>disabled</code>.</p> <p>When omitted, the default is <code>enabled</code>, signifying the server must verify the other server's certificate.</p> <p>When this parameter is <code>disabled</code>, the server establishes secure communication with the other server, but does not verify the server's identity.</p>

Table 47 SSL Parameters for Routes (Sheet 3 of 3)

Parameter	Description
ssl_verify_hostname	<p>Specifies whether the server must verify the name in the CN field of the other server’s certificate. The values for this parameter are enabled and disabled.</p> <p>When omitted, the default is enabled, signifying the server must verify the name of the connected host or the name specified in the ssl_expected_hostname parameter against the value in the server’s certificate. If the names do not match, the connection is rejected.</p> <p>When this parameter is disabled, the server establishes secure communication with the other server, but does not verify the server’s name.</p>
ssl_expected_hostname	<p>Specifies the name expected in the CN field of the other server’s certificate. If this parameter is not set, the default is the hostname of the other server.</p> <p>This parameter is relevant only when the ssl_verify_hostname parameter is enabled.</p>
ssl_ciphers	<p>Specifies a list of cipher suites, separated by colons (:).</p> <p>This parameter accepts both the OpenSSL name for cipher suites, or the longer descriptive names.</p> <p>For information about available cipher suites and their names, see Specifying Cipher Suites on page 281.</p>
ssl_rand_egd	<p>The path for the installed entropy gathering daemon (EGD), if one is installed. This daemon generates random numbers.</p>

Routed Topic Messages

A server forwards topic messages along routes only when the `global` property is defined for the topic; see `addprop topic` on page 168 and `create topic` on page 171.

Topic messages can traverse multiple hops.

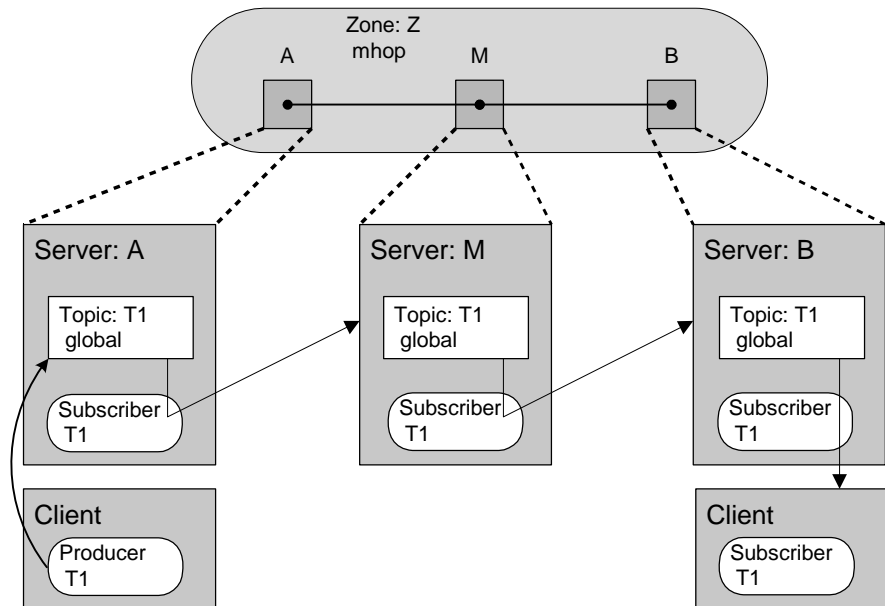
When a route becomes disconnected (for example, because of network problems), the forwarding server stores topic messages. When the route reconnects, the server forwards the stored messages.

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

Propagating Registered Interest

To ensure forwarding of messages along routes, servers propagate their topic subscriptions to other servers. For example, the top of Figure 25 depicts an enterprise with three servers—A, M and B—connected by routes in a multi-hop zone. The bottom of Figure 25 illustrates the mechanism at work within the servers to route messages from a producer client of server A, through server M, to server B and its subscriber client. Consider this sequence of events.

Figure 25 Routing: Propagating Subscribers



1. All three servers configure a global topic T1.
2. At bottom right of Figure 25, a client of server B creates a subscriber to T1.
3. Server B, registers interest in T1 on behalf of the client by creating an internal subscriber object.
4. Because a route connects servers M and B, server B propagates its interest in T1 to server M. In response, M creates an internal subscriber to T1 on behalf of server B. This subscriber ensures that M forwards (that is, delivers) messages from topic T1 to B. Server B behaves as a client of server M.
5. Similarly, because a route connects servers A and M, server M propagates its interest in T1 to server A. In response, A creates an internal subscriber to T1 on behalf of server M. This subscriber ensures that A forwards messages from topic T1 to M. Server M behaves as a client of server A.
6. When a producer client of server A sends a message to topic T1, A forwards it to M. M accepts the message on its topic T1, and forwards it to B. B accepts the message on its topic T1, and passes it to the client.

Subscriber Client Exit	If the client of server B creates a <i>non-durable</i> subscriber to T1, then if the client process exits, the servers delete the entire sequence of internal subscribers. When the client restarts, it generates a new sequence of subscribers; meanwhile, the client might have missed messages.
	If the client of server B creates a <i>durable</i> subscriber to T1, then if the client process exits, the entire sequence of internal subscribers remains intact; messages continue to flow through the servers in store-and-forward fashion. When the client restarts, it can consume all the messages that B has been storing in the interim.
Server Failure	In an active-active route between servers B and M, if B fails, then M retains its internal subscriber and continues to store messages for clients of B. When B reconnects, M forwards the stored messages.
	In an active-passive route configured on B, if B fails, then M removes its internal subscriber and does not store messages for clients of B—potentially resulting in a gap in the message stream. When B reconnects, M creates a new internal subscriber and resumes forwarding messages.
maxbytes	Combining durable subscribers with routes creates a potential demand for storage—especially in failure situations. For example, if server B fails, then server M stores messages until B resumes. We recommend that you set the <code>maxbytes</code> property of the topic (T1) on each server, to prevent unlimited storage growth (which could further disrupt operation).

Selectors for Routing Topic Messages

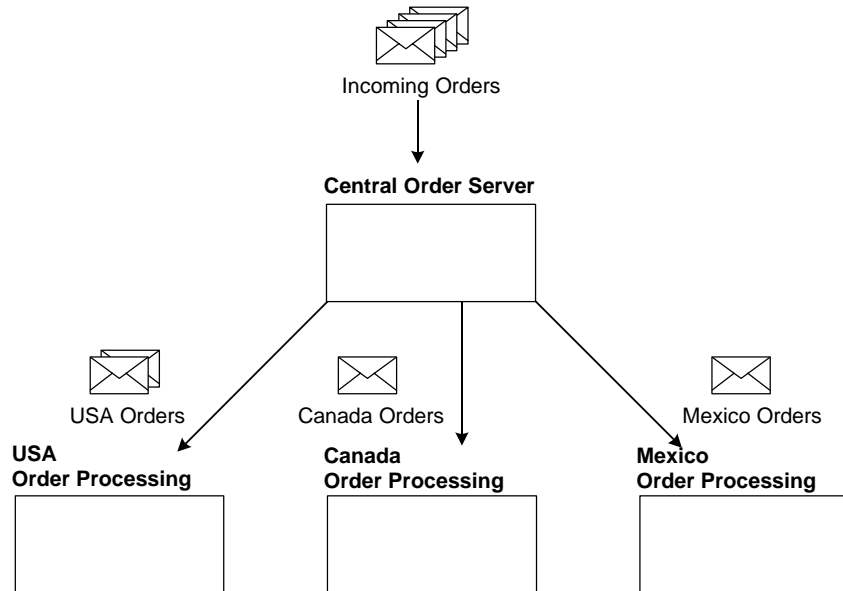
Motivation A server forwards a global topic message along routes to all servers with subscribers for that topic. When each of those other servers requires only a small subset of the messages, this policy could potentially result in a high volume of unwanted network traffic. You can specify *message selectors* on routes, to narrow the subset of topic messages that traverse each route.



Message selectors on routes are different from message selectors on individual subscribers, which narrow the subset of messages that the server delivers to the subscriber client.

Example Figure 26 illustrates an enterprise with a central server for processing customer orders, and separate regional servers for billing those orders. For optimal use of network capacity, we configure topic selectors so that each regional server gets only those orders related to customers within its region.

Figure 26 Routing: Topic Selectors, example



Specifying Selectors Specify message selectors for global topics as properties of routes. You can define these properties in two ways:

- Define selectors when creating the route (either in `routes.conf`, or with the administrator command `create route`).

- Manipulate selectors on an existing route (using the `addprop`, `setprop`, or `removeprop` administrator commands).

Syntax The message selector properties have the same syntax whether they appear in a command or in a configuration file:

```
incoming_topic=topicName selector="messageSelector"
outgoing_topic=topicName selector="messageSelector"
```



The terms *incoming* and *outgoing* refer to the perspective of the active server—where the route is defined.

topicName is the name of a global topic.

messageSelector is a message selector string. For detailed information about message selector syntax, see the documentation for class `Message` in *TIBCO Enterprise Message Service Java API Reference*.

Example Syntax In the example of Figure 26, an administrator might configure these routes on the central order server:

```
setprop route Canada outgoing_topic="orders" selector="country='Canada'"
setprop route Mexico outgoing_topic="orders" selector "country='Mexico'"
setprop route USA outgoing_topic="orders" selector="country='USA'"
```

Those commands would create these entries in `routes.conf`:

```
[Canada]
url=ssl://canada:7222
outgoing_topic=orders selector="country='Canada'"
...
[Mexico]
url=ssl://mexico:7222
outgoing_topic=orders selector="country='Mexico'"
...
[USA]
url=ssl://usa:7222
outgoing_topic=orders selector="country='USA'"
...
```

Symmetry `outgoing_topic` and `incoming_topic` are symmetric. Whether A specifies a route to B with `incoming_topic` selectors, or B specifies a route to A with `outgoing_topic` selectors, the effect is the same.

Active-Active Configuration In an active-active configuration, you may specify selectors on either or both servers. If you specify `outgoing_topic` selector `S1` for topic `T` on server `A`, and `incoming_topic` selector `S2` for `T` on server `B`, then the effective selector for `T` on the route from `A` to `B` is `(S1 AND S2)`.

See also *Active and Passive Routes* on page 311.

Wildcards You can specify wildcards in topic names. For each actual topic, the server uses logical AND to combine all the selectors that match the topic.

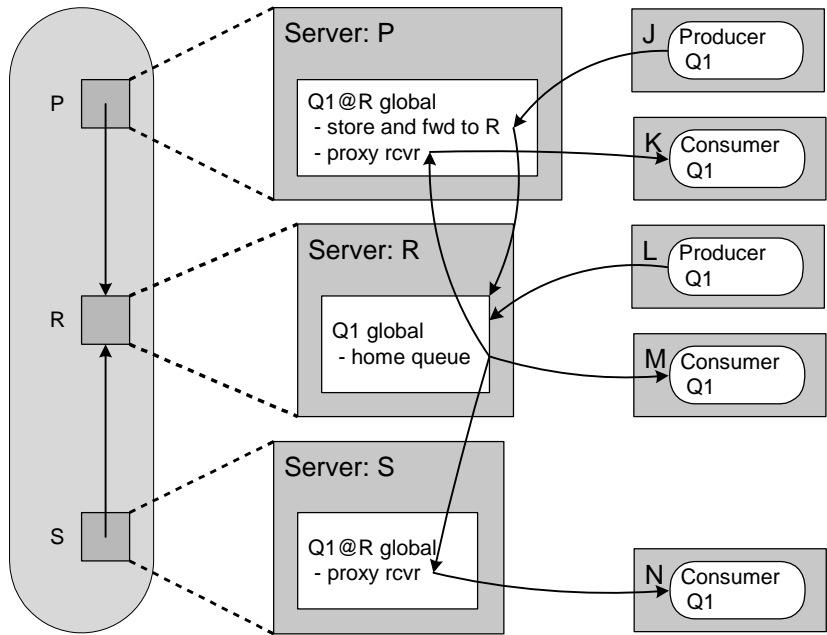
Routed Queues

With respect to routing, queues differ from topics in several respects:

- Servers route queue messages between the queue owner and adjacent servers.
- The concept of zones and hops does not apply to queue messages (only to topic messages).


The left side of Figure 27 depicts an enterprise with three servers—P, R and S—connected by routes. The remainder of Figure 25 illustrates the mechanisms that routes queue messages among servers (center) and their clients (right side).

Figure 27 Routing: Queues



Owner & Home Server R defines a global queue named Q1. R is the *owner* of Q1.

Servers P and S define *routed queues* Q1@R. This designation indicates that these queues depend upon and reflect their *home queue* (that is, Q1 on server R). Notice that the designation Q1@R is only for the purpose of configuration; clients of P refer to the routed queue as Q1.

Example	<p>When J sends a message to Q1, server P forwards the message to the home queue—Q1 on server R.</p> <p>Now the message is available to receivers on all three servers, P, R and S—although only one client can consume the message. Either Q1 on P receives it on behalf of K; or Q1 on S receives it on behalf of N; or M receives it directly from the home queue.</p>
Producers	<p>From the perspective of producer clients, a routed queue stores messages and forwards them to the home queue. For example, when J sends a message to Q1 on server P, P forwards it to the queue owner, R, which delivers it to Q1 (the home queue).</p> <p>The message is not available for consumers until it reaches the home queue. That is, client K cannot consume the message directly from server P.</p> <p>If server R fails, or the route connection from P to R fails, P continues to store messages from K in its queue. When P and R resume communication, P delivers the stored messages to Q1 on R.</p>
Consumers	<p>From the perspective of consumer clients, a routed queue acts as a proxy receiver. For example, when L sends a message to Q1 on server R, Q1 on P can receive it from R on behalf of K, and immediately gives it to K.</p> <p>If server P fails, or the route connection from P to R fails, K cannot receive messages from Q1 until the servers resume communication. Meanwhile, M and N continue to receive messages from Q1. When P and R resume communication, K can again receive messages through Q1 on P.</p>
Configuration	<p>You must explicitly configure each routed queue in <code>queues.conf</code>—clients cannot create routed queues dynamically.</p> <p>You may use the administration tool or API to configure routed queues; see <code>addprop queue</code> on page 167 and <code>create queue</code> on page 170.</p> <p>To configure a routed queue, specify the queue name and the server name of the queue owner; for example, on server P, configure:</p> <pre>Q1@R global</pre>
	<p>It is legal to use this notation even for the home queue. The queue owner recognizes its own name, and ignores the location designation (@R).</p>
	<p>It is illegal to configure a routed queue as <code>exclusive</code>.</p>
Browsing	<p>Queue browsers cannot examine routed queues. That is, you can create a browser only on the server that owns the home queue.</p>

Transactions



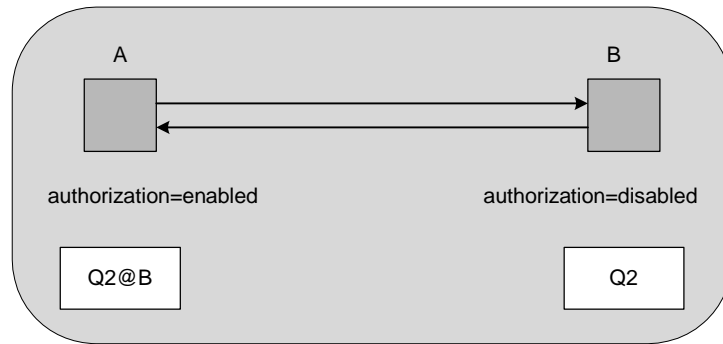
XA sessions do not support routed queues.

Routing and Authorization

User & Password

When a server's `authorization` parameter is enabled, other servers that actively connect to it must authenticate themselves by name and password, or by X.509 certificate.

Figure 28 Routing: Authorization



In Figure 28, servers A and B both configure active routes to one another.

- Because A enables authorization, A must configure a user named B, and B must configure a matching username and password to identify itself to A.
- However, because B disables authorization, A need not identify itself to B, and B need not configure a user named A.

ACL

When routing a secure topic or queue, servers consult the ACL specification before forwarding each message. The servers must grant one another appropriate permissions to send, receive, publish or subscribe.

For example, in Figure 28, Q2 messages flow from A to B if and only if server A grants receive permission to user B for Q2, and server B grants send permission to user A on Q2.

See Also Chapter 9, Authentication and Permissions, on page 199

Appendix A **Using the Samples**

This chapter describes working with the client sample files.

Topics

- *Starting Work with the Client Sample Files, page 328*
- *Publish and Subscribe Example, page 329*

Starting Work with the Client Sample Files

The sample files were designed to allow you to run TIBCO Enterprise Message Service with minimum start-up time and coding. The directory contains three sets of files:

- Client samples for TIBCO Enterprise Message Service implementation. (`jms/samples/java`)
- Examples created by Sun Microsystems as basic examples for JMS. (`jms/samples/java/sun`)
- Samples of interoperation of TIBCO Enterprise Message Service with TIBCO Rendezvous applications. (`jms/samples/java/tibrv`)

This section describes compiling and beginning to use the client samples.

Compiling the Sample Files

In order to compile and run the client samples you need to execute `setup.bat` (Windows) or `setup.sh` (UNIX) scripts, which are located in the sample files directory.

To run the client sample files:

1. Verify the setting of the `TIBEMS_ROOT` environment variable inside the `setup.bat` or `setup.sh` script file.
2. Open a command line or console window, and **change directory** to the `samples>client` subdirectory in the JMS folder.
3. Run the setup script.
4. Compile the samples by typing `javac -d. *.java` on the command line.

This will compile all the samples in the directory, except for those samples within the `sun` and `tibrv` subdirectories.

If the files compile successfully, the class files will appear in the `samples>client` subdirectory. If they do not compile correctly, an error message appears.

Publish and Subscribe Example

This section describes an example of publishing and subscribing, using the administration tool and the client samples.

Overview of the Example

In general, to run a client sample, you enter appropriate commands in the sample console window and the EMS administration tool. For more information on the sample clients, see the Readme in the sample folder. For further information on the administration tool, see [Starting the Administration Tool](#) on page 162 and [Command Listing](#) on page 167.

In this chapter, you will use the client samples. You will create a topic and then create two users for that topic. You will give one user permission to subscribe on that topic, and the other user permission to publish on that topic. Then you will publish two messages, and observe as they are received.

This example is simple and limited. For example, in a real application, the users would be connections to working applications, while in this example, users only send and receive messages. This example also does not attempt to include examples of all permissions, properties or commands, and it does not describe any of the queue commands. Nevertheless, the example can give a partial overview of the process.

Creating a Topic

In this section, you will create a topic. Several topics are included with the client samples. For this sample, however, you will create a new topic. You create topics in the EMS administration tool.



You must start the server and the administration tool before creating a topic. For information on running the server, refer to [Running the Server](#) on page 244. For information on starting the administration tool, refer to [Starting the Administration Tool](#) on page 162.

Using the Administration Tool

To begin using the administration tool, see [Starting the Administration Tool](#) on page 162.



On a computer running Windows, you can also start the administration tool from the Start menu, following the path **Programs>TIBCO Enterprise Message Service 4.3>Start EMS administration tool**.

When you have started the administration tool, you need to connect it to the EMS server.

To connect the EMS administration tool to the EMS server, execute one of the following commands:

- If you are using TIBCO Enterprise Message Service on a single computer, type **connect** in the command line of the Administration tool.

The screen will display: `connected to tcp://localhost:7222`.

It will then display the following prompt: `tcp://localhost:7222>`

- If you are using TIBCO Enterprise Message Service in a network, use the **connect server** command as follows:

```
connect [<server URL>] [<user-name>] [<password>]
```

For more information on this command, see [connect](#) on page 168.

Creating the Topic

Once you have connected the administration tool to the server, you can create a new topic.

To create a new topic:

1. Startup the administration tool, then enter the following command:

```
create topic foo
```

2. Enter the **commit** command to save the topic as a permanent topic.

For more information on the `create topic` command, refer to [create topic](#) on page 171. For more information on the `commit` command, see [commit](#) on page 168 and [autocommit](#) on page 168.

Adding the secure Property to the Topic

You will add the `secure` property to the topic. With the `secure` property added, only users who have been assigned a certain permission can perform the actions allowed by that permission. For example, only users with publish permission on the topic can publish, while other users cannot publish.

If the `secure` property is not added to a topic, all authenticated users have all permissions (publish, subscribe, create durable subscribers) on that topic.

For more information on the `secure` property, see the section about `secure` on page 36. For more information on topic permissions, see Chapter 9, Authentication and Permissions, on page 199.

To enable server authorization and add the `secure` property to a topic, do the following steps:

1. Start the server. For more information on starting the server, refer to Starting the Server on page 244.

2. Startup the administration tool.

3. Enable the authorization property by entering the following command:

```
set server authorization=enabled
```

The `authorization` property enables checking of permissions set on destinations.

4. Enter the following command to add the `secure` property to a topic named `foo`:

```
addprop topic foo secure
```

For more information on the `addprop topic` command, refer to `addprop topic` on page 168.

Creating Users

Using the client samples, you will create several users and give them various permissions to publish and subscribe to the topic `foo`.

The first step is creating the users.

Creating the Users

This section illustrates creating users with the administration tool.

To create two users:

1. Startup the administration tool, then enter the following command:

```
create user user1
```

The tool will display the message: `user user1 has been created.`

2. Enter the command:

```
create user user2
```

3. Enter the commit command.

You have now created two users.

For more information on the `create user` command, refer to create user on page 172.

Granting Permissions

In order to see how permissions affect the ability to publish and receive messages, you will give the two users various permissions on the topic `foo`. You will give publish permission to one user, subscribe permission to the second user, and both publish and subscribe to the third user.

To grant permissions to users on the topic `foo`:

1. Startup the administration tool, then enter the following commands:

```
grant topic foo user1 publish
grant topic foo user2 subscribe
```

2. Enter the `commit` command to save the permissions.

For more information on the `grant topic` command, refer to grant topic on page 174.

Next, you will use the client samples to publish and receive on topic `foo`.

Publishing and Subscribing

In previous sections, you have created the topic `foo`, assigned the property `secure` to that topic. You also made two users, and assigned permissions to publish and subscribe on `foo` to these users. In this section, you will use the client samples to publish and subscribe on `foo`.

To publish and subscribe on `foo`, you will use the client samples.

Running Client Samples

In order to run the client samples, you must give them commands from within the sample directory. The sample directory contains the compiled samples.

For more information on the samples, refer to the `readme` within the sample directory. For more information on compiling the samples, refer to Compiling the Sample Files on page 328.

Starting Subscribers

First, you will attempt to start both of your users as subscribers on `foo`. Because only one user has permission to subscribe on `foo`, only one of the users will actually start as a subscriber.

You will start the subscribers first, because the subscribers enable you to observe the messages being received when you start the publisher.

To start a subscriber:

1. Set a command line window and navigate to the folder containing the client samples.

2. At the prompt, enter: **setup**

Entering `setup` sets the environment and classpath, and returns you to the prompt.

3. After the prompt, enter:

```
java tibjmsTopicSubscriber -topic foo -user user1
```

However, in this example, `foo` is a secure topic, and `user1` has permission to publish, but not to subscribe. Therefore, you will receive an exception message including the statement:

```
Operation not permitted.
```

The window will then return to the prompt.

4. After the prompt, enter:

```
java tibjmsTopicSubscriber -topic foo -user user2
```

The screen will display a message showing that `user2` is subscribed to `foo`.

In this example, `foo` is a secure topic, and `user2` has permission to subscribe to `foo`.

The window does not return to the prompt, because the subscription is running, and no further action needs to be taken.

Starting the Publisher and Sending Messages

Setting up the publisher is very similar to setting up the subscriber. However, while the subscriber requires the name of the topic and the user, the publisher also requires messages.

To start the publisher:

1. Set a command line window and navigate to the folder containing the client samples.
2. At the prompt, enter: **setup**

Entering `setup` sets the environment and classpath, and returns you to the prompt.

3. After the prompt, enter:

```
java tibjmsTopicPublisher -topic foo -user user1 m1 m2
```

where `m1` and `m2` are messages.

The command line window containing the publisher will display a message stating that both messages have been published. The command line windows containing the subscriber will show the two messages being received.

After the messages are published, this window returns to the prompt for further message publishing.



Note that if you attempt to use the form:

```
java tibjmsTopicPublisher -topic foo -user user1
```

without adding the messages, you will see an error message, reminding you that you must have at least one message text.

Appendix B **Using TIBCO Hawk**

This appendix describes how to configure TIBCO Hawk so that it can be used to administer and monitor the TIBCO Enterprise Message Service server.

For more information about TIBCO Hawk, see the TIBCO Hawk documentation.

Topics

- *Overview of Server Management With TIBCO Hawk, page 336*
- *Installing the Classes, page 337*
- *Method Description, page 341*

Overview of Server Management With TIBCO Hawk

TIBCO Hawk is a tool for monitoring and managing applications and operating systems. TIBCO Enterprise Message Service provides classes for administering the EMS server. Table 48 describes the provided classes.

Table 48 TIBCO Enterprise Message Service classes in TIBCO Hawk

Class	Description
<code>com.tibco.tibjms.admin.hawk.HawkListener</code>	Monitoring methods that allow you to view the status of topics, queues, routes, and other items on the TIBCO Enterprise Message Service server.
<code>com.tibco.tibjms.admin.hawk.HawkController</code>	<div>Management methods for shutting down the TIBCO Enterprise Message Service server and performing other administrative functions.</div> <div>This class contains all <code>HawkListener</code> monitoring methods as well.</div>

If you wish to monitor and manage the server, you only need the `HawkController` class. If you wish to only monitor the server, you can use the `HawkListener` class.

To use TIBCO Hawk to manage the TIBCO Enterprise Message Service server, you must load the classes provided into the TIBCO Hawk agent. Once the classes are loaded, methods for managing the EMS server are available in the TIBCO Hawk display.

This appendix details how to install the provided classes into the TIBCO Hawk agent and the methods available for monitoring and managing the TIBCO Enterprise Message Service server.

Installing the Classes

Installing the provided classes is different for UNIX and Windows platforms. The following sections detail how to install the TIBCO Enterprise Message Service management classes into the TIBCO Hawk agent for each platform.



These instructions are specific to TIBCO Hawk Release 4.1.0 or later. Earlier versions of TIBCO Hawk have a different mechanism for adding plugins. Refer to your TIBCO Hawk documentation for details on installing plugins, if you are using an earlier version of TIBCO Hawk.

Windows Installation

To install the provided classes for use in a TIBCO Hawk agent running on a Windows platform, perform the following:

1. Locate the `tibemsadmin.hma` file in the TIBCO Enterprise Message Service installation directory under the `samples\admin\hawk` subdirectory and copy it into your TIBCO Hawk plugins directory.

Usually, a TIBCO Hawk plugins directory is located in `c:\tibco\hawk\plugins`.

2. When using Hawk earlier than 4.5, locate `jms.jar` and `tibjms.jar` in the `clients\java` subdirectory, and copy them into the TIBCO Hawk plugins directory.
3. For all Hawk versions, locate `tibjmsadmin.jar` in the `clients\java` subdirectory, and copy it into the TIBCO Hawk plugins directory.
4. Open the TIBCO Hawk Configuration Utility and make certain the plugins directory is set to the location where you have installed TIBCO Hawk plugins. To set the plugins directory, click the Agent tab, then set the Plugins Directory field to the location where the plugins are located.

For more information about using the TIBCO Hawk Configuration Utility, see *TIBCO Hawk Installation and Configuration*.

5. Navigate to your plugins directory and open the `tibemsadmin.hma` file in a text editor.
6. Specify the TIBCO Hawk microagent class you wish to use in the `<classname>` element. You can use either the `HawkListener` class if you only want to monitor the server, or you can specify the `HawkController` class if you want to monitor and manage the server.

7. Specify the username and password and server URL to use to connect to the TIBCO Enterprise Message Service server in the appropriate <arg> elements. See Table 49 on page 339.

For example:

```
<arguments>
  <arg>-user</arg>
  <arg>admin</arg>
  <arg>-password</arg>
  <arg>admin_pass</arg>
  <arg>-server</arg>
  <arg>tcp://server1.yourcompany.com:7222</arg>
  <arg>-timeout</arg>
  <arg>5</arg>
</arguments>
```

You should use specify the predefined admin user or a user that is a member of the \$admin group.

8. Restart the TIBCO Hawk agent service. See the TIBCO Hawk documentation for more information about restarting the service.

UNIX Installation

To install these classes for use in a TIBCO Hawk Agent running on a UNIX platform, perform the following procedure:

1. Locate the `tibemsadmin.hma` file in the TIBCO Enterprise Message Service installation directory under the `samples/hawk` subdirectory and copy it into your TIBCO Hawk plugins directory.

Usually, a TIBCO Hawk plugins directory is located in `/usr/tibco/hawk/plugins`.

2. When using Hawk earlier than 4.5, locate `jms.jar` and `tibjms.jar` in the `clients/java` subdirectory, and copy them into the TIBCO Hawk plugins directory.
3. For all Hawk versions, locate `tibjmsadmin.jar` in the `clients/java` subdirectory, and copy it into the TIBCO Hawk plugins directory.
4. Edit the TIBCO Hawk `hawkagent.cfg` file and specify the `-hma_plugin_dir` option to include the directory where your TIBCO Hawk plugins are located.

For more information about editing TIBCO Hawk configuration files on UNIX, see *TIBCO Hawk Installation and Configuration*.

5. Navigate to your plugins directory and open the `tibemsadmin.hma` file in a text editor.

6. Specify the TIBCO Hawk microagent class you wish to use in the <classname> element. You can use either the HawkListener class if you only want to monitor the server, or you can specify the HawkController class if you want to monitor and manage the server.
7. Specify the username and password and server URL to use to connect to the TIBCO Enterprise Message Service server in the appropriate <arg> elements. See Table 49 on page 339.

For example:

```
<arguments>
  <arg>-user</arg>
  <arg>admin</arg>
  <arg>-password</arg>
  <arg>admin_pass</arg>
  <arg>-server</arg>
  <arg>tcp://server1.yourcompany.com:7222</arg>
  <arg>-timeout</arg>
  <arg>5</arg>
</arguments>
```

You should use specify the predefined admin user or a user that is a member of the \$admin group.

Parameters

Table 49 TIBCO Hawk MicroAgent Parameters (Sheet 1 of 2)

Parameter	Description
-user	The MicroAgent identifies itself with this user name and password when it connects to the EMS server.
-password	
	When absent, the default user name is admin. When absent, the default password is the empty string.
-user	To use an encrypted password, specify this pair. As the value for -encryptedPassword, supply the output you obtain by running the Hawk utility program tibhawkpassword (which encrypts your password).
-encryptedPassword	
-server	The MicroAgent connects to the EMS server at this URL (host computer and port). When absent, the default is tcp://localhost:7222.

Table 49 TIBCO Hawk MicroAgent Parameters (Sheet 2 of 2)

Parameter	Description
-timeout	Limits the time (in seconds) that the MicroAgent waits for the EMS server to respond to queries. Acceptable values are in the range [5, 3600]. When absent, the default is 60.

Method Description

The TIBCO Hawk classes have several methods for managing and monitoring a TIBCO Enterprise Message Service server. These methods correspond to commands you can issue in the administration tool.

Table 50 lists the methods of each class and the corresponding `tibemsadmin` command for the method. The table also lists the page where you can find more information about each command in Chapter 8, Using the Administration Tool, on page 161.

Table 50 TIBCO Hawk method names (Sheet 1 of 3)

TIBCO Hawk Agent Method Name	tibemsadmin Command	Page
com.tibco.tibjms.admin.hawk.HawkListener Methods		
<code>getMethods()</code>	This method returns the list of methods that this TIBCO Hawk class can perform.	—
<code>getServerInfo()</code>	<code>show server</code>	194
<code>getNumConnections()</code>	Returns the number of connections.	—
<code>getConnections</code>	<code>show connections</code>	184
<code>getUsers()</code>	<code>show users</code>	197
<code>getQueues(String regexp)</code>	<code>show queue</code> <code>show queues</code> (shows information from both commands for each queue) You can specify a queue name or a pattern to return all matching queue names.	191
<code>getRoutes()</code>	<code>show route</code> <code>show routes</code> (shows information from both commands for each route)	192
<code>getTopics(String regexp)</code>	<code>show topic</code> <code>show topics</code> (shows information from both commands for each topic) You can specify a topic name or a pattern to return all matching topic names.	195

Table 50 TIBCO Hawk method names (Sheet 2 of 3)

TIBCO Hawk Agent Method Name	tibemsadmin Command	Page
getDurables(String regexp)	show durables You can specify a topic name or a pattern to return all matching durable subscriptions.	188
getConsumers()	show stat consumers	194
getProducers()	show stat producers	194
getListenPorts()	This method returns the list of ports the TIBCO Enterprise Message Service server is configured to listen on.	—
getCMLedgerInfo()	show rvcmlledger	193
getTransports()	show rvcmllistener	194
getTransport()	show rvcmllistener Shows the name and subject of the specified RVCML listener.	194
isRunning()	Check whether the server is reachable by attempting to connect to it. (Afterward, this method breaks the test connection.)	—
com.tibco.tibjms.admin.hawk.HawkController Methods (also contains all HawkListener methods)		
getMethods()	This method returns the list of methods that this TIBCO Hawk class can perform.	—
shutdown()	shutdown	197
purgeDurable(String name, String clientID)	purge durable Specify the name of the durable subscription and the client ID associated with the durable subscription (client ID can be null).	176
purgeQueue(String name)	purge queue Specify the name of the queue to purge.	176
purgeTopic(String name)	purge topic Specify the name of the topic to purge.	176

Table 50 TIBCO Hawk method names (Sheet 3 of 3)

TIBCO Hawk Agent Method Name	tibemsadmin Command	Page
rotateLog()	rotatelog	177

Appendix C **Monitor Messages**

This appendix lists all topics on which the server publishes messages for system events. The message properties for messages published on each topic are also described. See Monitoring Server Events on page 233 for more information about monitor topics and messages.

Topics

- *Description of Monitor Topics, page 346*
- *Description of Topic Message Properties, page 349*

Description of Monitor Topics

Table 51 describes each monitor topic.

Table 51 Monitor topics (Sheet 1 of 3)

Topic	Message Is Published When...
<code>\$sys.monitor.admin.change</code>	The administrator has made a change to the configuration.
<code>\$sys.monitor.connection.connect</code>	A user attempts to connect to the server.
<code>\$sys.monitor.connection.disconnect</code>	A user connection is disconnected.
<code>\$sys.monitor.connection.error</code>	An error occurs on a user connection.
<code>\$sys.monitor.consumer.create</code>	A consumer is created.
<code>\$sys.monitor.consumer.destroy</code>	A consumer is destroyed.
<code>\$sys.monitor.flow.engaged</code>	Stored messages rise above a destination's limit, engaging the flow control feature.
<code>\$sys.monitor.flow.disengaged</code>	Stored messages fall below a destination's limit, disengaging the flow control feature.
<code>\$sys.monitor.limits.connection</code>	Maximum number of hosts or connections is reached.
<code>\$sys.monitor.limits.queue</code>	Maximum bytes for queue storage is reached.
<code>\$sys.monitor.limits.server</code>	Server memory limit is reached.
<code>\$sys.monitor.limits.topic</code>	Maximum bytes for durable subscriptions is reached.
<code>\$sys.monitor.producer.create</code>	A producer is created.
<code>\$sys.monitor.producer.destroy</code>	A producer is destroyed.
<code>\$sys.monitor.queue.create</code>	A dynamic queue is created.
<code>\$sys.monitor.route.connect</code>	A route connection is attempted.
<code>\$sys.monitor.route.disconnect</code>	A route connection is disconnected.
<code>\$sys.monitor.route.error</code>	An error occurs on a route connection.

Table 51 Monitor topics (Sheet 2 of 3)

Topic	Message Is Published When...
<code>\$sys.monitor.route.interest</code>	A change in registered interest occurs on the route.
<code>\$sys.monitor.server.info</code>	The server sends information about an event; for example, a log file is rotated.
<code>\$sys.monitor.server.state</code>	The server state changes; for example, an administrator shuts down the server.
<code>\$sys.monitor.topic.create</code>	A dynamic topic is created.
<code>\$sys.monitor.tx.action</code>	A local transaction commits or rolls back.
<code>\$sys.monitor.xa.action</code>	An XA transaction commits or rolls back.

Table 51 Monitor topics (Sheet 3 of 3)

Topic	Message Is Published When...
<code>\$sys.monitor.<D>.<E>.<destination></code>	<p>A message is handled by a destination. The name of this monitor topic includes two qualifiers (<i>D</i> and <i>E</i>) and the name of the destination you wish to monitor.</p> <p><i>D</i> signifies the type of destination and whether to include the entire message:</p> <ul style="list-style-type: none">• <code>T</code> — topic, include full message (as a byte array) into each event• <code>t</code> — topic, do not include full message into each event• <code>Q</code> — queue, include full message (as a byte array) into each event• <code>q</code> — queue, do not include full message into each event <p><i>E</i> signifies the type of event:</p> <ul style="list-style-type: none">• <code>r</code> for receive• <code>s</code> for send• <code>a</code> for acknowledge• <code>*</code> for all event types <p>For example, <code>\$sys.monitor.T.r.corp.News</code> is the topic for monitoring any received messages to the topic named <code>corp.News</code>. The message body of any received messages is included in monitor messages on this topic. The topic <code>\$sys.monitor.q.*.corp.*</code> monitors all message events (send, receive, acknowledge) for all queues matching the name <code>corp.*</code>. The message body is not included in this topic's messages.</p> <p>The messages sent to this type of monitor topic include a description of the event, information about where the message came from (a producer, route, external system, and so on), and optionally the message body, depending upon the value of <i>D</i>.</p> <p>See Monitoring Messages on page 233 for more information about message monitoring.</p>

Description of Topic Message Properties

Table 52 describes the properties that monitor topic messages can contain. Each monitor message can have a different set of these properties.

Table 52 Message properties (Sheet 1 of 6)

Property	Contents
conn_connid	Connection ID of the connection that generated the event.
conn_ft	Whether the client connection is a connection to a fault-tolerant server.
conn_hostname	Hostname of the connection that generated the event.
conn_ssl	Whether the server connection uses the SSL protocol.
conn_ssl2	Whether the client connection uses the SSL protocol.
conn_type	Type of connection that generated the event. This property can have the following values: <ul style="list-style-type: none"> • Admin • Topic • Queue • Generic • Route • FT (connection to fault-tolerant server) • Unknown
conn_username	User name of the connection that generated the event.
conn_xa	Whether the client connection is an XA connection.

Table 52 Message properties (Sheet 2 of 6)

Property	Contents
event_action	<p>The action that caused the event. This property can have the values listed in Table 53 on page 355.</p> <ul style="list-style-type: none">• connect (connection attempted)• accept (connection accepted)• disconnect (connection disconnected)• interest (registered interest for a route)• create (something created)• delete (something deleted)• modify (something changed)• add (user added to a group)• remove (user removed from a group)• grant (permission granted)• revoke (permission revoked)• purge (topic, queue, or durable subscriber purged)• commit (transaction committed)• rollback (transaction rolled back)• rotateatelog (log file rotated)• receive (message posted into destination)• send (message sent by server to another party)• acknowledge (message is acknowledged)
event_class	<p>The type of monitoring event (that is, the last part of the topic name without the <code>\$sys.monitor</code>).</p> <p>For message monitoring, the value of this property is always set to <code>message</code>.</p>
event_reason	<p>The reason the event occurred (usually an error). The values this property can have are described in Table 54.</p>
event_route	<p>For routing, the route that the event occurred on.</p>

Table 52 Message properties (Sheet 3 of 6)

Property	Contents
message_bytes	When the full message is to be included for message monitoring, this field contains the message as a byte array. You can use the createFromBytes method (in the various client APIs) to recover the message.
mode	<p>Message delivery mode. This values of this property can be the following:</p> <ul style="list-style-type: none"> • persistent • non_persistent • reliable
msg_seq	Message sequence number.
msg_id	Message ID.
msg_timestamp	Message timestamp.
msg_expiration	Message expiration.
replyTo	Message JMSReplyTo.
rv_reply	Message RV reply subject.
source_id	ID of the source object.
source_name	<p>Name of the source object involved with the event. This property can have the following values:</p> <ul style="list-style-type: none"> • XID (global transaction ID) • message_id • connections (number of connections) • unknown (unknown name) • Any server property name • the name of the user, or anonymous

Table 52 Message properties (Sheet 4 of 6)

Property	Contents
source_object	<div>Source object that was involved with the event. This property can have the following values:</div> <ul style="list-style-type: none">producerconsumertopicqueuepermissionsdurableservertransactionusergroupconnectionmessagejndinamefactoryfilelimits (a limit, such as a memory limit)routetransport
source_value	Value of source object.
stat_msgs	Message count statistic for producer or consumer.
stat_size	Message size statistic for producer or consumer.
target_admin	Whether the target object is the admin connection.
target_created	Time that the consumer was created (in milliseconds since the epoch).
target_dest_name	Name of the target destination

Table 52 Message properties (Sheet 5 of 6)

Property	Contents
<code>target_dest_type</code>	Type of the target destination.
<code>target_durable</code>	Name of durable subscriber when target is durable subscriber.
<code>target_group</code>	Group name that was target of the event
<code>target_hostname</code>	Hostname of the target object.
<code>target_id</code>	ID of the target object.
<code>target_name</code>	<p>Name of the object that was the target of the event. This property can have the following values:</p> <ul style="list-style-type: none"> • <code>XID</code> (global transaction ID) • <code>message_id</code> • <code>connections</code> (number of connections) • <code>unknown</code> (unknown name) • Any server property name • the name of the user, or <code>anonymous</code>
<code>target_nolocal</code>	NoLocal flag when target is durable subscriber.

Table 52 Message properties (Sheet 6 of 6)

Property	Contents
target_object	<p>The general object that was the target of the event. This property can have the following values:</p> <ul style="list-style-type: none">• producer• consumer• topic• queue• permissions• durable• server• transaction• user• group• connection• message• jndiname• factory• file• limits (a limit, such as a memory limit)• route• transport
target_selector	Selector when the target is a consumer.
target_subscription	Subscription of the target object when target is durable subscriber.
target_url	URL of the target object.
target_username	Username of the target object.
target_value	Value of the object that was the target of the event, such as the name of a topic, queue, permission, and so on.

Table 53 Event Action Property Values (Sheet 1 of 2)

Event Action Value	Description
accept	connection accepted
acknowledge	message is acknowledged
add	user added to a group
admin_commit	administrator manually committed an XA transaction
admin_rollback	administrator manually rolled back an XA transaction
commit	transaction committed
connect	connection attempted
create	something created
delete	something deleted
disconnect	connection disconnected
flow_engaged	stored messages rise above a destination's limit, engaging the flow control feature
flow_disengaged	stored messages fall below a destination's limit, disengaging the flow control feature
interest	registered interest for a route
modify	something changed
grant	permission granted
purge	topic, queue, or durable subscriber purged
receive	message posted into destination
remove	user removed from a group
resume	administrator resumed a route
revoke	permission revoked

Table 53 Event Action Property Values (Sheet 2 of 2)

Event Action Value	Description
rollback	transaction rolled back
rotate_log	log file rotated
send	message sent by server to another party
suspend	administrator suspended a route
txcommit	administrator manually committed a local transaction
txrollback	administrator manually rolled back a local transaction
xaccommit	an application committed an XA transaction (2-phase)
xaccommit_1phase	an application committed an XA transaction (1-phase)
xastart	an application started a new XA transaction
xastart_join	an application has joined (that is, added) a resource to an existing transaction
xastart_resume	an application resumed a suspended XA transaction
xaend_fail	an application ended an XA transaction, indicating failure
xaend_success	an application ended an XA transaction, indicating success
xaend_suspend	an application suspended an XA transaction
xaprepare	an application prepared an XA transaction
xarecover	an application called recover (to get a list of XA transactions)
xarollback	an application rolled back an XA transaction

Table 54 Event Reason Property Values (Sheet 1 of 2)

Event Reason Value	Description
backup_connected	The fault-tolerant backup server has connected.
backup_disconnected	The fault-tolerant backup server has disconnected.
bridge	Message posted to destination as result of bridging.
closed	Connection was closed.
consumer	For message monitoring, this value signifies a message was sent or acknowledged by a consumer. For all other cases, this value signifies a dynamic topic or queue created for a consumer.
cycle	Cyclic route created.
disabled	Feature not enabled.
duplicate	Duplicate, such as route, global queue or topic.
error	Connection disconnected due to error.
exceeded	Limit exceeded.
export	Message exported to a transport.
import	Message imported from a transport.
invalid_name	Name not valid, such as route name.
invalid_password	Invalid password provided.
not_authorized	Not authorized to perform action.
not_connected	Could not establish connection.
not_found	Something was expected, but not found.
producer	For message monitoring, this value signifies a message was posted by a producer. For all other cases, this value signifies a dynamic topic or queue created for a producer.

Table 54 Event Reason Property Values (Sheet 2 of 2)

Event Reason Value	Description
reconnect_active	Connection active.
reconnect_unknown	Connection unknown.
rotatelog	Log file rotated.
route	For message monitoring, this value signifies a message was sent or received from a route. For all other cases, this value signifies a dynamic topic or queue created for a route.
shutdown	Server was shut down.
standby	Server in standby mode.
terminated	Connection was terminated.

Appendix D **Error and Status Messages**

This appendix lists all possible error messages that the server can output, alphabetized by category.

Key to this Appendix

Category	The category indicates the general class of error. This appendix is alphabetized by category.
Description	The description explains the error category in more detail.
Resolution	The resolution indicates possible recovery actions that administrators should consider.
Errors	These strings represent all instances of the error, as they appear in EMS server code. Some categories have many error instances; others have only one. These strings can include formatting characters.

Error and Status Messages

Category	A durable consumer was found in the store file for a route that does not exist
Description	On server startup a durable consumer was found in the store file for a route that is not listed in the routes.conf file. This happens if the routes.conf file is manually edited.
Resolution	Make routing changes via administration tools.
Errors	Discarding durable '%s' for route '%s' because the route does not exist.

Category	Admin command failed
Description	An admin tool or program using the admin API attempted an operation that failed for the given reason.
Resolution	The admin tool or admin API provides the failure reason. The user of the tool or API should examine the error and correct the syntax, parameter or configuration that is causing the failure.
Errors	%s: create %s failed: conflicting zone: existing consumer has a different zone %s: create %s failed: detected duplicate durable subscription [%s] for topic [%s]. %s: create %s failed: illegal to use wildcard %s [%s]. %s: create %s failed: invalid %s [%s]. %s: create %s failed: invalid session id=%d. %s: create %s failed: invalid syntax of %s [%s]. %s: create %s failed: invalid temporary %s [%s]. %s: create %s failed: not allowed to create dynamic %s [%s].

Category	Backup server '%s' disconnected
Description	Lost connection with the backup fault-tolerant server.
Resolution	Determine if the backup server is running. If it is running, check for a network partition.

Errors	Backup server '%s' disconnected.
Category	Bad or missing value for command line parameter
Description	An invalid value was supplied for a command line parameter.
Resolution	Change the value of the named parameter to an acceptable value; for information about tibemsd command line parameters, see EMS documentation.
Errors	'%s' requires an integer argument. '%s' requires a positive integer argument. '%s' requires a string argument.
Category	Banners and debug traces
Description	Banner and debug traces
Resolution	Not applicable
Errors	%s: Message swapping has been %s Invalid session for route configuration. Expired %d message%s. [%s@%s]: rejected connect from route: invalid password %s: purged durable '%s' %s: %s %s '%s' permissions on %s '%s': %s %s: create %s failed: durable creation access denied for %s [%s]. Async Recs: max=%d avg=%.2f min=%d Process Id: %d Server activating on failure of '%s'. ldap_search_ext_s(%0x, %s, %s, %s) Flow Stall Recovery Timer: to recover stall of %s on route from %s, recovery count = %d Error, filter '%s' contains an illegal type substitution character, only %%%s is allowed Allocating sync storage to minimum %s, please wait.

Rendezvous Certified Advisory: %s

LDAP response resulting from checking if an entry is a member of a dynamic group:'

ignoring route '%s' at '%s', route user does not exist.

Created %s transport '%s'

Send recover request for routed queue flow stall for queue %s

Removing routed topic consumer '%s'

License has been activated.

Hostname: %s

Evaluation Software Notice: remaining uptime is %d hours %d minutes.

[%s@%s]: rejected connect from route: implicit route already exists

LDAP response resulting from getting attributes for group '%s':

ldap_parse_reference: %s

Storage Location: '%s'.

Search reference: %s

Route Recover Interval is %u seconds.

Route connect error: route has no zone setting

SS: Deleting existing GMD file.

LDAP error: %s

Clean all flow stalls for route to server %s: %s

%s: shutdown server

Reading configuration from '%s'.

%s: Maximum statistics memory set to unlimited

Configuration warning: file=%s, line=%d: illegal to specify both '%s' and '%s', ignoring '%s'

Recovered flow stalled consumer for destination: %s:%s

%s: revoked all %s permissions on %s '%s'

Error sending routing information to '%s'.

Send recover request

LDAP Cache: Adding static group '%s' to group membership for '%s'

Lazy Dels: max=%d avg=%.2f min=%d

%s: created rvcmlistener transport '%s' name '%s' dest '%s'

ERROR: file=%s, line=%d: server name is too long,

Route '%s' connected to url '%s' with zone '%s:%s'.

[%s@%s]: rejected connect from route: %s

Configuration warning: file=%s, line=%d: Use of Rendezvous Bridge via tibrv_... parameters has been deprecated. This feature is subject to removal in the next release of this product. Please convert your configuration to utilize transports defined in transports.conf configuration file.

Rendezvous %s %s enabled (RV %s).

Error in ldap_search_ext_s: %s

Server is re-entering standby mode.

Statistics database memory now below limit

SS: Destroying SmartSockets transport %s

Created file '%s'

Restored routed topic consumer for '%s'

LDAP message resulting from searching for groups:

Adding routed topic consumer for '%s'

Subscriber %s for topic '%s' exceeded memory limit of % PRINTF_LL_FMT d bytes.

Refrained from removing configured durable '%s'

Sync Recs: max=%d avg=%.2f min=%d

SS: Unsubscribe from '%s' tport = %s

Recovered %d pending connection%s.

%s: Message ID tracking has been %s

LDAP message resulting from searching for users:

SS: Imported message on tport='%s', subject='%s', reply='%s'.

Clean flow stall for consumers of destination %s:%s

ldap_search_s(%0x, %s, %s, %s, [NULL])

%s:%s queue browser failed: illegal to use wildcard queue [%s]

There should be only one consumer reaching %s, but %d found

%s:%s queue browser failed: cannot browse [%s] because it is a routed queue.

Detected IP interface: %s (%s)

Clear (Non-IO) flow stalled on dest %s:%s from route of %s

Error sending routing information to %s, send failed

%s: console_trace updated: '%s'

%s: consumed_msg_hold_time updated: '%d'

%s: Server SSL password has been changed

Authorization is disabled.

SSL connect: using certificate username '%s'.

SSL reset to TCP for connID=% PRINTF_LLFMT d, user='%s'

Configuration warning: file=%s, line=%d: invalid trace option '%s' is ignored

Server is now active.

(NON-IO) Flow stalled on dest %s from route of %s

Dump of static cache:

Administrator group not found, created with default member.

Received exception on route '%s': '%s'

%s: log_trace updated: '%s'

ldap_search_s(%x, %s, %s, %s, [%s,%s,%s,%s,%s,NULL])

EXPIRE: -

Category	Basic initialization failed
Description	tibemsd was unable to start.
Resolution	Correct the configuration or startup parameters and restart.
Errors	Unable to add admin user into admin group: error=(%d) %s Fault tolerant activation has to be greater than 2x heartbeat Fault Tolerant configuration error, can't create loop. Fault tolerant connection failed, fault tolerant mode not supported on '%s'. Fault tolerant heartbeat has to be greater than 0 Initialization failed due to errors in configuration. Initialization failed due to errors in SSL.

Initialization failed due to errors with transports.

Initialization failed. Exiting.

Initialization has failed. Exiting.

Initialization of thread pool failed (%s). Exiting.

Startup aborted.

Server failed to read configuration.

Initialization failed: database file '%s' not found.

Category	Commit failed due to prior failure or after fault-tolerant switch
Description	A warning message indicating that the commit of a client application's transaction failed either because there were earlier errors when processing this transaction or because the transaction was started on the primary server prior to a fault-tolerant failover.
Resolution	The client application should retry the transaction.
Errors	Commit failed due to prior failure or after fault-tolerant switch.

Category	Compaction failed
Description	Compaction of the store file failed.
Resolution	The most likely cause of this error is running out of memory. Shut down tibemspd and see remedies for Out of memory.
Errors	Compaction failed (%s). Please shutdown and restart tibemspd

Category	Configured durable differs from stored one
Description	The durables configuration file specifies a durable with a given name and client identifier with attributes that are different from the identically named durable found in the meta.db file.
Resolution	Correct the durables configuration file to match the durable defined in the meta.db file or administratively delete the durable and re-define it.

Errors	Configured durable '%s' differs from durable in storage, storage version used.
Category	Create of global routed topic failed: not allowed to create dynamic topic
Description	A server received an interest notification from another server that does not match the allowed topics in its configuration.
Resolution	This only is printed when the trace includes ROUTE_DEBUG. If the server's topic definitions are as expected, this statement can be ignored or remove the ROUTE_DEBUG trace specification to prevent printing.
Errors	Create of global routed topic failed: not allowed to create dynamic topic [%s].
Category	Create of routed queue failed: not allowed to create dynamic queue
Description	A warning indicating that a tibemsd with a route to this daemon has a queue configured to be global but this daemon does not permit the creation of that queue dynamically.
Resolution	Add the specified queue or a pattern that includes it to this daemon if you want the queue to be accessible from this daemon, otherwise the warning can be ignored.
Errors	Create of routed queue failed: not allowed to create dynamic queue [%s].
Category	Database record damaged
Description	An error occurred reading one of the tibemsd store files.
Resolution	Send details of the error and the situation in which it occurred to TIBCO Support.
Errors	Invalid destination for message. No destination information for consumer. Persisted message possibly corrupt. Server failed to recover state.
Category	dead code, not reachable
Description	tibemsd encountered a SmartSockets error.

Resolution	See SmartSockets documentation for details of what the error means and how to remedy it.
Errors	<p>Unable to initialize connection, SSL username error.</p> <p>LDAP authentication failed for user '%s', status = %d</p> <p>LDAP authentication failed for user '%s', no password provided</p> <p>LDAP authentication failed for user '%s', no password provided</p> <p>user_auth SYSTEM: The server must run as root for SYSTEM password support.</p>
Category	Duplicate message detected
Description	Warning generated when tibemsd receives a message with a message id that matches another message's message id.
Resolution	Only seen when message id tracking is enabled.
Errors	Detected duplicate %s message, messageID='%s'
Category	Error in configuration file
Description	The server encountered an invalid configuration statement in the specified configuration file on the specified line.
Resolution	Examine the appropriate configuration file and correct the syntax error.
Errors	<p>Configuration warning: file=%s, line=%d: route '%s' does not have a user configured for authorization.</p> <p>SSL Configuration error: file=%s, line=%d: invalid certificate file name, unknown extension or invalid encoding specification</p> <p>Configuration error: file=%s, line=%d: illegal to specify exclusive for routed queue</p> <p>Configuration error: file=%s, line=%d: ignored alias '%s' for %s '%s' because such alias already exists</p> <p>Configuration error: file=%s, line=%d: both tibrv_export and tibrvcm_export are specified, ignoring tibrv_export</p> <p>Configuration error: file=%s, line=%d: ignoring transport '%s' in %s list, transport not found</p>

Configuration error: file=%s, line=%d: multiple bridge entries for the same destination '%s' are not allowed.

Configuration error: file=%s, line=%d: Ignoring durable, name cannot start with \$sys.route, use route property instead.

Configuration error: file=%s, line=%d: Rendezvous transport not specified for Rendezvous CM transport '%s'

Configuration error: file=%s, line=%d: ignoring invalid max connections in the line, reset to unlimited

Configuration error: file=%s, line=%d: value of %s out of range, reset to default

Configuration error: file=%s, line=%d: unable to create %s '%s': invalid destination name, invalid parameters or out of memory

Configuration error: file=%s, line=%d: value of db_pool_size too big or less than allowed minimum, reset to default value of %d bytes

Configuration error: file=%s, line=%d: Ignoring durable, route does not allow clientid, selector or nlocal.

Configuration error: file=%s, line=%d: unable to process selector in route parameters, error=%s

Configuration error: file=%s, line=%d: both tibrv_import and tibrvcm_import are specified, ignoring tibrv_import

Configuration error: file=%s, line=%d: ignored route '%s' because route represents route to this server.

Configuration error: file=%s, line=%d: ignoring invalid topic selector specifications in route parameters

Configuration error: file=%s, line=%d: value of max_msg_memory less than allowed, reset to %dMB

Configuration error: file=%s, line=%d: ignored alias '%s' for factory because such alias already exists

Configuration error: file=%s, line=%d: specified value below allowable minimum. Resetting value store_minimum to 8M.

Configuration error: file=%s, line=%d: specified value below allowable minimum. Resetting value store_minimum_sync to 8M.

Configuration error: file=%s, line=%d: invalid certificate file name, unknown extension or invalid encoding specification

Configuration error: file=%s, line=%d: ignored route '%s' because route has invalid zone information.

Configuration error: file=%s, line=%d: ignored route '%s' because route with such name or URL already exists.

Configuration error: file=%s, line=%d: value of msg_pool_size invalid or too big or less than allowed minimum of %d, reset to default value of %d

SSL Configuration error: file=%s, line=%d: invalid private key file name, unknown extension or invalid encoding specification

Configuration conflict: file=%s, line=%d: value of msg_pool_block_size already set at line=%d. Ignoring msg_pool_block_size.

Configuration error: file=%s, line=%d: bridge has no targets, unable to process

Configuration error: file=%s, line=%d: Illegal to specify routed queue as a bridge source

Configuration error: file=%s, line=%d: client_trace error: %s

Configuration error: file=%s, line=%d: %s

Configuration error: file=%s, line=%d: duplicate specification of transport type

Configuration error: file=%s, line=%d: duplicate value

Configuration error: file=%s, line=%d: Ignoring durable, duplicate of earlier entry.

Configuration error: file=%s, line=%d: Ignoring durable, name is invalid.

Configuration error: file=%s, line=%d: Ignoring durable, name is missing or invalid.

Configuration error: file=%s, line=%d: Ignoring durable, topic is invalid.

Configuration error: file=%s, line=%d: Ignoring durable, topic is missing or invalid.

Configuration error: file=%s, line=%d: error in the bridge description, unable to proceed.

Configuration error: file=%s, line=%d: error in permissions

Configuration error: file=%s, line=%d: error in the transport description, unable to proceed.

Configuration error: file=%s, line=%d: errors in line, some options may have been ignored

Error: unable to add bridge specified in file=%s, line=%d. Error=%s

Configuration error: file=%s, line=%d: Unable to create destination defined by the bridge source

Unable to create Rendezvous Certified transport '%s' because it references undefined Rendezvous transport '%s'

Configuration error: file=%s, line=%d: failed to create ACL entry, reason=%s

Unable to export message to SmartSockets.

Use fsync error: file=%s, line=%d: invalid property value

Use fsync (min disk) error: file=%s, line=%d: invalid property value

exit_on_nonretryable_disk_error: file=%s, line=%d: invalid boolean property value

consumed_msg_hold_time: file=%s, line=%d: invalid property value

Fault tolerant reread error: file=%s, line=%d: invalid property value

Fault standby lock check error: file=%s, line=%d: invalid property value

Configuration error: file=%s, line=%d: ignored unknown permission '%s'

Configuration error: file=%s, line=%d: ignoring duplicate %s '%s' specified earlier

Configuration error: file=%s, line=%d: ignoring duplicate transport name '%s' in %s list

Configuration error: file=%s, line=%d: ignoring duplicate user

Configuration error: file=%s, line=%d: ignoring errors in permission line

Configuration error: file=%s, line=%d: ignoring invalid connect attempt count

Configuration error: file=%s, line=%d: ignoring invalid connect attempt delay

Configuration error: file=%s, line=%d: ignoring invalid disk stat period

Configuration error: file=%s, line=%d: ignoring invalid entry syntax

Configuration error: file=%s, line=%d: ignoring invalid factory load balancing metric

Configuration error: file=%s, line=%d: ignoring invalid ft activation in the line

Configuration error: file=%s, line=%d: ignoring invalid ft heartbeat in the line

Configuration error: file=%s, line=%d: ignoring invalid ft reconnect timeout in the line

Configuration error: file=%s, line=%d: ignoring invalid line

Configuration error: file=%s, line=%d: ignoring invalid line in factory parameters

Configuration error: file=%s, line=%d: ignoring invalid line in route parameters

Configuration error: file=%s, line=%d: ignoring invalid line: invalid syntax in the line

Configuration error: file=%s, line=%d: ignoring invalid reconnect attempt count

Configuration error: file=%s, line=%d: ignoring invalid reconnect attempt delay

Configuration error: file=%s, line=%d: ignoring invalid value of %s

Configuration error: file=%s, line=%d: ignoring invalid value in the line

Configuration error: file=%s, line=%d: ignoring unknown property '%s'

Configuration error: file=%s, line=%d: ignoring unrecognized property '%s'

Configuration error: file=%s, line=%d: ignoring user out of group context

Configuration error: file=%s, line=%d: illegal to use predefined name %s

Configuration error: file=%s, line=%d: Invalid clientid value

Configuration error: file=%s, line=%d: invalid value of db_pool_size, reset to default of %d bytes

Configuration error: file=%s, line=%d: invalid line syntax or line out of order

Configuration error: file=%s, line=%d: invalid line syntax or line out of order

Configuration error: file=%s, line=%d: invalid value of max memory, reset to unlimited

Configuration error: file=%s, line=%d: invalid value of max_msg_memory, reset to unlimited

Configuration error: file=%s, line=%d: invalid property value

Configuration error: file=%s, line=%d: invalid property value, reset to default.

Configuration error: file=%s, line=%d: invalid password

Configuration error: file=%s, line=%d: invalid value of reserve_memory, reset to zero

Configuration error: file=%s, line=%d: invalid value of route_recover_interval, reset to default %d

Configuration error: file=%s, line=%d: Invalid selector value

Configuration error: file=%s, line=%d: invalid syntax of %s, unable to continue.

Configuration error: file=%s, line=%d: invalid transport parameter '%s'

Configuration error: file=%s, line=%d: invalid transport type '%s'

Configuration error: file=%s, line=%d: invalid trace_client_host value

Configuration error: file=%s, line=%d: invalid value of %s, reset to unlimited

Configuration error: file=%s, line=%d: invalid value, reset to no minimum.

Configuration error: file=%s, line=%d: invalid value '%s'

Configuration error: file=%s, line=%d: invalid value of '%s'

Configuration error: file=%s, line=%d: invalid value of %s

Configuration error: file=%s, line=%d: invalid value of %s, reset to 256MB

Configuration error: file=%s, line=%d: invalid value of %s, reset to default

Configuration error: file=%s, line=%d: line too long, ignoring it

Configuration error: file=%s, line=%d: maximum number of listen interfaces reached.

Configuration error: file=%s, line=%d: multiple principals specified, line ignored

Configuration error: file=%s, line=%d: multiple targets specified, line ignored

Configuration error: file=%s, line=%d: multiple targets specified, line ignored

Configuration error: file=%s, line=%d: out of memory, unable to create Rendezvous transport

Configuration error: file=%s, line=%d: no permissions found in acl entry

Configuration error: file=%s, line=%d: no target found in acl entry

Configuration error: file=%s, line=%d: %s '%s' not found

No topic exists for configured durable '%s%s%s'.

Configuration error: file=%s, line=%d: no valid user or group found in acl entry

Configuration conflict: file=%s, line=%d: Overriding value of msg_pool_size already set at line=%d.

Configuration warning: file=%s, line=%d: parameter '%s' is deprecated

Configuration error: file=%s, line=%d: value of reserve_memory too small, reset to 16MB

Configuration error: file=%s, line=%d: ignoring invalid line in route parameters: invalid zone type, too long

Configuration error: file=%s, line=%d: ignoring invalid line in route parameters: zone name exceeding %d bytes

Routing Configuration error: file=%s, line=%d: invalid property value

Configuration warning: file=%s, line=%d: ignoring rvcmlistener, duplicate

Configuration error: file=%s, line=%d: ignoring rvcmlistener, first token is invalid

Configuration error: file=%s, line=%d: ignoring rvcmlistener, invalid destination

Configuration error: file=%s, line=%d: ignoring rvcmlistener, second token is invalid

Configuration error: file=%s, line=%d: ignoring rvcmlistener, third token is invalid

Configuration error: file=%s, line=%d: ignoring rvcmlistener, wildcards are not permitted

SmartSockets configuration directory name is too long.

SmartSockets file '%s' not found.

SSL Configuration error: file=%s, line=%d: duplicate value

SSL Configuration error: file=%s, line=%d: invalid value of DH key size.

SSL Configuration error: file=%s, line=%d: invalid property value

SSL Configuration error: file=%s, line=%d: invalid renegotiate size value

SSL Configuration error: file=%s, line=%d: invalid renegotiate size value, minimum is %dKb

SSL Configuration error: file=%s, line=%d: invalid renegotiate value, minimum is %d (in seconds)

Configuration error: file=%s, line=%d: syntax error in the line, ignoring

Configuration error: file=%s, line=%d: syntax errors in line, line ignored

Topic '%s' not valid in configured durable '%s'.

Configuration error: file=%s, line=%d: Unrecognized attribute

Configuration error: file=%s, line=%d: user '%s' not found, ignoring

Configuration error: file=%s, line=%d: value is invalid or less than minimum %d, reset to 0

Configuration error: file=%s, line=%d: value less than allowed minimum, reset to 0

Configuration error: file=%s, line=%d: value of %s less than allowed minimum of %dKB, reset to unlimited

Category	Error writing commit request, errors already occurred in this transaction
Description	A client application's attempt to commit a transaction failed because the server encountered an error during an operation associated with the transaction.
Resolution	Examine previous error statements to determine the cause of the operation failure and correct that before attempting the transaction again.

Errors	Error writing commit request, errors already occurred in this transaction.
<hr/>	
Category	Error writing configuration file
Description	tibemsd was unable to update one of its configuration files following a configuration change.
Resolution	Check that the user that started the tibemsd has permission to change the configuration files and that there is sufficient disk space on the device.
Errors	Error occurred saving acl information Error occurred saving bridges information Error occurred saving durables information Error occurred saving factories information Error occurred saving file '%s' Error occurred saving group information Error occurred saving %s information Error occurred saving main configuration file '%s' Error occurred saving routes information Error occurred saving tibrvcm information Error occurred while updating main configuration file '%s'. Configuration has not been saved. Error occurred writing bridges into file. Error occurred writing destination '%s' into file Error occurred writing factory into file. Error occurred writing group '%s' into file Error occurred writing into the file '%s'. Error occurred writing route into file. I/O error occurred saving group information I/O error occurred saving bridge information I/O error occurred saving group information I/O error occurred saving route information

I/O error occurred writing into file '%s'

Category	Error writing to store file
Description	tibemsd was unable to write data to one of its store files.
Resolution	Ensure that the directory containing the store files is mounted and accessible to the tibemsd, and that there is free space available on the device
Errors	<p>Error writing xa prepare request, %s.</p> <p>Failed writing block data to '%s': %s</p> <p>Failed writing message to '%s': I/O error or out of disk space.</p> <p>Failed writing purge state for queue '%s': I/O error or out of disk space.</p> <p>Failed writing purge state for topic consumer: I/O error or out of disk space.</p> <p>Exception trying to create ack record, %s.</p> <p>Exception trying to create confirm record, %s.</p> <p>Exception trying to create message from store: %s</p> <p>Exception trying to create transaction record.</p> <p>Exception trying to create valid messages record, %s.</p> <p>Exception trying to export message to RV.</p>

Category	Failed to open TCP port
Description	tibemsd was unable to open the tcp port.
Resolution	Shutdown process that is using the port or change the value of the 'listen' parameter in the server's tibemsd.conf file to a port that is not in use.
Errors	Binding connection to TCP port %d failed:%d (%s).

Category	Fault tolerant reconnect timeout is set to a large value of %d seconds
Description	Warning that fault tolerant reconnect timeout is set to a large number of seconds.
Resolution	Consider reducing the timeout unless it is important that the newly active server maintains state for clients that do not reconnect following a failover.

Errors	<p>Fault Tolerant error, can't create connection to '%s'.</p> <p>Fault tolerant reconnect timeout is set to a large value of %d seconds.</p>
Category	File access error
Description	tibemspd was unable to open the specified file.
Resolution	Check that the path name is correct and the directory exists, the user that started tibemspd has permission to read the specified directory and path, the file exists if it isn't one that the tibemspd can create, the file is not being used by another tibemspd or some other process.
Errors	<p>Configuration file '%s' not found.</p> <p>Failed to create file '%s'</p> <p>failed to open file '%s'.</p> <p>failed to open log file '%s'.</p> <p>Failed to read message from store.</p> <p>Failed to rename file %s into %s: %s</p> <p>Unable to open metadata file '%s', error '%s'.</p> <p>Unable to open metadata file '%s', file may be locked.</p> <p>Unable to open store file '%s', error '%s'.</p> <p>Unable to open store file '%s', file may be locked.</p> <p>Unable to preallocate async storage file '%s'.</p> <p>Unable to preallocate sync storage file '%s'.</p> <p>I/O error occurred reading from the file '%s'.</p> <p>I/O error occurred reading from the file '%s'.</p> <p>Exiting on non-retryable disk error: %d</p> <p>Exception trying to read message from store.</p>
Category	In comment field
Description	Warning indicating that tibemspd was attempting to reestablish delivery of messages across a route to another tibemspd but was unable to find the connection for that route.

Resolution	Either reduce the tibemspd's memory requirement by consuming messages or removing messages from its queues, or increase the amount of memory available to the tibemspd by shutting down other processes on the machine or increasing the machine's memory.
Errors	<p>Unable to send route resume message</p> <p>Invalid client version record detected.</p> <p>Invalid version record detected.</p>
Category	Internal error that should be status-driven
Description	The server detected an internal consistency.
Resolution	Send the error statement and a description of the environment to TIBCO Support.
Errors	<p>**Error** unable to process message, error = %s</p> <p>Admin user not found during initialization</p> <p>Asynchronous producer is sending a message into non-failsafe destination. This is not yet supported.</p> <p>Error bridging transacted data message, '%s'.</p> <p>Error processing xa commit request, %s.</p> <p>Error processing xa end - transaction marked ROLLBACKONLY, %s.</p> <p>Error processing xa prepare request, %s.</p> <p>Error processing xa rollback request, %s.</p> <p>Error decoding sequence data in xa rollback request.</p> <p>Unable to create internal session</p> <p>Problem setting flow stall recover message on route queue:%s: %s</p> <p>Failed to handle connection init: %s.</p> <p>Problem trying to recover routed consumer for queue %s: setting recover message. Error: %s</p> <p>Failed to send the flow stall recover request: %s.</p> <p>Unable to handle transacted data message, '%s'.</p> <p>Unable to handoff connection init message: %s.</p> <p>Unable to initialize fault tolerant connection, remote server returned '%s'</p>

Unable to process producer message, failed to add sender name, error=%s.

Unable to process sequence for message.

Unable to send recover ack on flow stall: %s

Handling of route flow stall recovery request from %s failed: unable to get message property %s: %s

Handling of route flow stall recovery request failed: Unable to get message properties:%s

Failed to send acknowledge to the stall recover request of server %s, will try later.
Error: %s

failed to send recover ack on stalled flow: invalid consumer

Exception creating connection.

Exception creating purge record.

Exception creating session.

Exception creating zone.

Exception creating zone: adding zone to state.

Exception in startup, exiting.

Exception preparing message for client send.

Exception restoring persisted message, %s.

Exception sending flow recover acknowledge

Exception sending routing information to %s.

Exception sending session init response

Exception trying to initialize connection.

Exception trying to initialize connection, can't send response.

Exception trying to initialize route.

Exception trying to process message, '%s'.

Exception trying to process message from store.

Category	Invalid connection
Description	Warning indicating that tibemspd was attempting to reestablish delivery of messages across a route to another tibemspd but was unable to find the connection for that route.

Resolution	Either reduce the tibemspd's memory requirement by consuming messages or removing messages from its queues, or increase the amount of memory available to the tibemspd by shutting down other processes on the machine or increasing the machine's memory.
Errors	Recovery flow stall for destination %s failed: invalid route connection
Category	Invalid destination
Description	An application is attempting to use a destination name that is not valid.
Resolution	Alter application code to use an acceptable destination name.
Errors	<p>%s: create %s failed: Not permitted to use reserved queue [%s].</p> <p>%s: %s failed: illegal to use wildcard %s [%s].</p> <p>%s: %s failed: not allowed to create dynamic %s [%s].</p>
Category	Invalid listen specification
Description	The server could not parse the listen parameter in the tibemspd.conf file
Resolution	Correct the listen parameter to match the form [protocol]://[url] as specified in the manual.
Errors	<p>Invalid listen specification: '%s'.</p> <p>Invalid request to create temporary destination.</p>
Category	Invalid session
Description	tibemspd received a request that referred to a session that doesn't currently exist.
Resolution	Send details of the error and the situation in which it occurred to TIBCO Support.
Errors	<p>Cannot find session for ack</p> <p>Cannot find session for ack range</p> <p>%s: destroy %s failed: invalid session id=%d.</p> <p>Unable to destroy session, invalid session.</p> <p>Invalid session in commit request.</p>

Invalid session in commit transaction record.
Invalid session in recover request.
Invalid session in rollback request.
Invalid session in xa end request.
Invalid session in xa start request.

Category	LDAP error - should always display LDAP error
Description	An attempt to authenticate a client's userid and password using the external LDAP server failed.
Resolution	Examine the error code printed by the messaging server and consult the manual for the external LDAP server.
Errors	filter '%s' contains an illegal type substitution character, only %%%s is allowed filter '%s' contains too many occurrences of %%%s, max allowed is: %d filter '%s' too long, max length is %d characters invalid search scope: %s LDAP Configuration error: file=%s, line=%d: invalid property value LDAP is not present LDAP search resulted %d hits. ldap_url_parse failed, returned: %d lookup of group '%s' produced incorrect or no results missing LDAP URL missing %s parameter zero entries returned from getting attributes for group '%s':

Category	LICENSE WARNING
Description	The server detected a violation of its license.
Resolution	This error only occurs with the evaluation version of the server or in an embedded form. To correct this error either replace your evaluation version with a production version or contact the vendor who supplied the embedded version.

Errors	License violation: %s.
<hr/>	
Category	Missing configuration
Description	An essential attribute has not been configured.
Resolution	Change the tibemsd.conf file so that a value for the attribute is provided.
Errors	Configuration error with metadata database. Configuration error with storage databases.
<hr/>	
Category	Missing transaction
Description	A client application attempted to change the state of a transaction that the tibemsd does not have in its list of current transactions.
Resolution	Check tibemsd trace logs to see if the transaction had been committed or rolled back by an administrator, if not then check the client code to see if it or its transaction manager are calling the transaction operations in the correct order.
Errors	Cannot find transaction referred to in commit request. Cannot find transaction referred to in commit transaction record. Cannot find transaction referred to in prepare request. Cannot find transaction referred to in rollback request. Cannot find transaction referred to in rollback transaction record. Cannot find transaction referred to in xa commit request. Cannot find transaction referred to in xa prepare request. Cannot find transaction referred to in xa rollback request. Received prepare request for transaction already prepared. Cannot find transaction referred to in xa start request. Cannot process xa end for non-existent transaction.
<hr/>	
Category	Out of memory
Description	The server failed to allocate memory as it was attempting to perform an operation.

Resolution	Check how much memory the server process is using according to the operating system. Compare this with how much memory and swap space the host actually has. If there are sufficient memory and swap space, check the operating system limits on the server process to determine if this is the cause. If the limits are set to their maximum and this error occurs, reduce the load on this server by moving some topics and queues to another server.
Errors	<p>%s trying to recreate persistent message.</p> <p>%s to create message from store.</p> <p>Error during routed queue configuration, can not create routed queue consumer</p> <p>Could not initialize monitor</p> <p>Error: out of memory processing admin request</p> <p>Error during route configuration, can not create routed queue consumer</p> <p>Unable to create admin group: out of memory during initialization</p> <p>Error: unable to create alias for %s '%s': no memory</p> <p>Error: unable to create alias: out of memory</p> <p>Unable to create import event for %s '%s' on transport '%s'</p> <p>Unable to create internal connection, error=(%d) %s</p> <p>Unable to create internal connection: out of memory during initialization</p> <p>Error: unable to create %s '%s': no memory</p> <p>Error: unable to create route while parsing file=%s, line=%d.</p> <p>Unable to create SmartSockets subscriber on transport '%s', %s '%s': out of memory</p> <p>Unable to create temporary destination, out of memory</p> <p>Failed to create reserve memory. Exiting.</p> <p>Failed writing message to '%s': No memory for operation.</p> <p>Unable to process message imported on transport '%s'.</p> <p>Fault Tolerant configuration, no memory!</p> <p>Fault Tolerant error, no memory.</p> <p>LDAP initialization failed.</p> <p>No memory.</p> <p>No memory authenticating user '%s'</p> <p>No memory authenticating via LDAP</p>

Out of memory while building admin response message
Out of memory while building JNDI response message
Out of memory creating global import event on transport '%s'
Out of memory creating import event for %s '%s' on transport '%s'
Out of memory creating SS transport %s
No memory creating stalled flows in destination
Out of memory during initialization
Out of memory exporting SS message.
Out of memory: unable to process SS message type on export
No memory for creating connection.
No memory generating dynamic route durable.
Out of memory importing SS message
No memory in IO thread to create pool.
Out of memory while parsing bridges file
Out of memory while parsing factories file
Out of memory while parsing routes file
No memory performing routing operation.
Out of memory processing %s on %s '%s'
Out of memory processing administrative request
Out of memory processing message tracing
No memory processing purge record.
No memory while processing route interest
Out of memory processing transports
Out of memory processing transports configuration
Out of memory reading configuration.
Out of memory restoring routed consumer
Out of memory sending monitor message.
No memory sending topic routing information.
No memory trying to add message to dead queue.
No memory trying to add message to system.

No memory trying to cleanup route.

No memory to create ack record.

No memory to create client connection

No memory to create configured durable '%s%s%s'.

No memory to create configured durables

No memory to create confirm record.

No memory to create connection.

No memory to create consumer.

No memory trying to create destination.

No memory to create destination for consumer.

No memory to create destination for message.

No memory to create destination for producer.

No memory trying to create global topic destination.

No memory to create message from store.

No memory trying to create message producer.

No memory to create producer.

No memory trying to create queue browser.

No memory trying to create response message.

No memory to create routed consumer

No memory to create routed queue consumers

No memory trying to create routed queue destination.

No memory trying to create routed tmp queue destination.

No memory to create session.

No memory trying to create tmp destination for consumer.

No memory trying to create transaction.

No memory to create transaction record.

No memory to create valid messages record.

No memory to create zone.

No memory trying to export message to RV.

No memory trying to export message to SS.

No memory trying to import message from RV%s.
 No memory trying to import message from RVCM.
 No memory trying to import message from SS.
 No memory trying to initialize connection.
 No memory trying to initialize route connection.
 No memory trying to parse configured durable.
 No memory trying to process data message.
 No memory trying to process queue message.
 No memory to process route interest
 No memory to process SSL renegotiation request.
 No memory trying to process system request.
 No memory trying to process topic consumer.
 No memory trying to process topic message.
 No memory trying to process xa end.
 No memory trying to read message from store.
 No memory trying to recover routed consumer.
 No memory trying to recover route stall.
 No memory trying to recover route stall, will try again.
 No memory to restore messages.
 No memory to restore prepared transactions.
 No memory trying to retrieve for queue browser.
 No memory trying to send recover/rollback response.
 No memory trying to send recover/rollback response.
 out of memory trying to send topic interest to routes
 No memory to set clientID for connection.
 No memory trying to setup queue route configuration
 No memory trying to setup route configuration
 No memory trying to setup topic route configuration
 Route recovery of destination %s on route from %s will fail: No memory

Route recovery of destination %s on route from %s will fail: No memory to create timer

Route recovery of destination %s on route from %s will fail: No memory to record state

Failed to initialize OpenSSL environment: out of memory

Exception trying to create ack record for prepared transaction, no memory.

Exception trying to create ack record, no memory.

Category	Protocol error, incorrect XID in XA request
Description	The tibemsd received an XA End instruction from the third party Transaction Manager which referred to a different transaction from the one currently in use by the session.
Resolution	Report this to the your Transaction Manager vendor.
Errors	Incorrect xid in xa end request.

Category	Protocol error, transaction in incorrect state
Description	A client application's attempt to start an XA transaction failed because the transaction already exists and is not in the correct state.
Resolution	This error is most likely caused by an external transaction manager that allowed two separate client applications to use the same XA transaction identifier (XID). Consult the manual for the transaction manager or report this to the transaction manager vendor.
Errors	Cannot process xa start for a session when another transaction is already active on that session
	Cannot process xa start with TMNOFLAGS when the transaction is already active.

Category	Protocol message format error
Description	tibemsd received a message with either missing or incomplete data.
Resolution	Send details of the error and the situation in which it occurred to TIBCO Support.

Errors Unable to confirm session, invalid request.

 Unable to create consumer, invalid destination.

 Unable to init session, invalid request.

 Unable to process msg for export.

 Unable to recover consumer, invalid request.

 Unable to recover consumer, invalid session.

 Unable to serve the flow stall recover request from server %s, invalid request.

 Unable to start consumer, invalid consumer

 Unable to server the flow stall recover request from server %s, invalid consumer.

 Unable to unsubscribe consumer, invalid client request.

 %s: %s failed: illegal to use %s [%s] in standby mode.

 Invalid destination information for producer.

 Invalid flag in xa end request.

 Invalid flag in xa start request.

 Invalid request to delete temporary destination.

 Invalid request to delete temporary destination: not owner connection.

 Invalid xid in commit request.

 Invalid xid in commit transaction record.

 Invalid xid in prepare request.

 Invalid xid in rollback request.

 Invalid xid in rollback transaction record.

 Invalid xid in xa commit request.

 Invalid xid in xa end request.

 Invalid xid in xa prepare request.

 Invalid xid in xa rollback request.

 Invalid xid in xa start request.

 Malformed routed message

 Problem decoding sequence data in confirm.

 Problem decoding sequence data in xa end.

 %s:%s queue browser failed: queue name is missing in request message

	Received admin request with replyTo not set
	Received JNDI request with replyTo not set.
	Received unexpected message type %d
<hr/>	
Category	Protocol sequence error
Description	A non-embedded java client is attempting to connect to a tibemsd that is part of an embedded JMS environment.
Resolution	Reconfigure the client to connect to a fully licensed tibemsd.
Errors	Invalid client connect detected. No closure.
<hr/>	
Category	Rejected attempt to connect via SSL to TCP port
Description	A client application attempted to connect to the server's TCP port using the SSL protocol.
Resolution	Change the client application's URL from ssl to tcp or change the server's listen parameter from tcp to ssl. To activate a change of the server listen parameter requires a restart of the server.
Errors	Rejected attempt to connect via SSL to TCP port
<hr/>	
Category	Rejected attempt to connect via TCP to SSL port
Description	A client application attempted to connect to the server's SSL port using the TCP protocol.
Resolution	Change the client application's URL from tcp to ssl or change the server's listen parameter from ssl to tcp. To activate a change of the server listen parameter requires a restart of the server.
Errors	Rejected attempt to connect via TCP to SSL port
<hr/>	
Category	rejected connect from route: invalid cycle in route

Description The multi-hop route support of the server does not support configuring a cycle. However, it detected a configuration that would create a cycle.

Resolution Remove one of the routes that creates the cycle.

Errors [%s@%s]: rejected connect from route: invalid cycle in route: %s
 Illegal, route to '%s' creates a cycle. Terminate the connection
 Illegal, route to '%s' creates a cycle.

Category Rendezvous transport error

Description tibemsd encountered a Rendezvous error.

Resolution See Rendezvous documentation for details of what the error means and how to remedy it.

Errors Unable to confirm RVCN message. %s
 Unable to create inbox for import event for %s '%s' on transport '%s'
 Unable to create Rendezvous Certified transport '%s': %s
 Unable to create Rendezvous Certified transport '%s' because unable to create Rendezvous transport '%s'
 Unable to create Rendezvous transport '%s': %s
 Unable to create TIBCO Rendezvous Certified Listener for %s '%s' on transport '%s': %s
 Failed to confirm RVCN message: %s.
 Failed to disallow Rendezvous Certified Message listener '%s': %s
 Unable to export topic message, error=%s.
 Unable to handoff confirm RVCN message: %s.
 Unable to pre-register certified listener '%s' on transport '%s': %s
 Rendezvous send failed on transport '%s', error='%s'

Category Restoring consumer failed

Description Seen when tibemsd starts up and detects that the zone for a route as specified in routes.conf has been changed.

Resolution	Either delete the route or change its zone back and restart the tibemsd.
Errors	Restoring consumer failed: Conflicting zone for route to [%s]: The route was initially zone %s type %s, but now %s type %s. Zone change not allowed while there are durable subscribers. Please delete the route first and create new one.
Category	Running on reserve memory
Description	Warnings indicating that the tibemsd has run out of memory and is now using its reserve memory
Resolution	Either reduce the tibemsd's memory requirement by consuming messages or removing messages from its queues, or increase the amount of memory available to the tibemsd by shutting down other processes on the machine or increasing the machine's memory.
Errors	Running on reserve memory, ignoring new message Running on reserve memory, no more send requests accepted.
Category	SmartSockets transport error
Description	tibemsd encountered a SmartSockets error.
Resolution	See SmartSockets documentation for details of what the error means and how to remedy it.
Errors	Unable to create SmartSockets subscriber on transport '%s': failed to convert %s '%s', error=%s Unable to import SmartSockets message on transport %s: failed to convert subject '%s', error=%d Unable to import SmartSockets message on transport %s: failed to convert reply subject '%s', error=%s Unable to export EMS message into SmartSockets on transport '%s'. Failed to convert subject '%s', error=%s. Unable to export EMS message into SmartSockets on transport '%s'. Failed to convert reply subject '%s', error=%s. Error translating EMS message body into SS message. Status=%s Error translating EMS message headers into SS message. Status=%s Error translating EMS message properties into SS message. Status=%s

Unable to confirm SS message. %s
 Unable to connect to SmartSockets RTserver via transport: '%s': %d - %s
 Unable to register GMD failure callback: '%s': %d - %s
 Unable to create open callback on transport: '%s': %d - %s
 Unable to create SS callback for %s '%s' on transport '%s' SS error: %s
 Unable to create SS message type on export
 Unable to create SmartSockets subscriber for %s '%s' on transport '%s', error: %s
 Unable to create SmartSockets transport '%s': %d - %s
 Failed to confirm SS message.
 Failed to create SmartSockets transport %s
 Unable to handoff confirm SS message: %s.
 Unable to import SS message. Error=%d, %s.
 Unable to import SS message, error retrieving delivery mode.
 Unable to import SS message, error retrieving number of fields.
 Unable to initialize SmartSockets transport '%s': error=%d: %s
 Unable to set SmartSockets Dispatcher for transport: '%s': %d - %s
 Unable to set SS message type on export
 Unable to set Username/Password for SmartSockets transport '%s': %d - %s
 Unable to import SmartSockets message on transport %s: failed to retrieve SS subject.
 SS Subject CB destroy Failed: for '%s' on transport '%s' SS error: %s
 SS Subject CB lookup Failed: for '%s' on transport '%s' SS error: %s
 SmartSockets TipcMsgSetDeliveryMode failed, '%s'
 SmartSockets TipcMsgSetLbMode failed, '%s'
 SmartSockets TipcSrvConnFlush failed, '%s'
 SmartSockets TipcSrvConnMsgSend failed, '%s'
 SS Unsubscribe failed: for '%s' on transport '%s' SS error: %s
 GMD delivery failed on transport '%s', SS message seq=%d, reason='%s' for process '%s'
 Unable to process undelivered SS GMD message, can not register EMS message, error='%s', tport='%s', GMD seq=%d

Unable to process undelivered SS GMD message, can not add to undelivered EMS queue, error='%s', tport='%s', GMD seq=%d

Unable to process undelivered SS GMD message, failed to build EMS message, error='%s', tport='%s', GMD seq=%d

Unable to convert undelivered SS GMD message into EMS message, error='%s', tport='%s', GMD seq=%d

Category	SSL initialization failed
Description	The server failed attempting to initialize the OpenSSL library.
Resolution	Examine the OpenSSL error and the EMS User's Guide chapter describing the use of SSL.
Errors	Failed to process ft ssl password Failed to process ssl password Ignoring SSL listen port %s Failed to initialize SSL: can not load certificates and/or private key and/or CRL file(s) Failed to initialize OpenSSL environment: error=%d, message=%s. Failed to initialize SSL. Error=%s Failed to initialize SSL: unable to obtain password Failed to initialize SSL: server certificate not specified. Failed to initialize SSL: server private key not specified.

Category	System call error, should be errno-driven
Description	A low-level system function has failed.
Resolution	Report the error to your system administrator and ask them to remedy the problem.
Errors	Accept() failed: too many open files. Please check per-process and system-wide limits on the number of open files. Accept() failed: %d (%s) Cannot retrieve user name of the current process.

Client connection not created, socket failed.

Could not obtain hostname

Could not resolve hostname '%s'. Possibly default hostname is not configured properly while multiple network interfaces are present.

Unable to listen for connections: %d (%s).

Unable to open socket for listening: %d (%s).

Category	Unnecessary or duplicate message
Description	tibemsd received a message with either missing or incomplete data.
Resolution	Send details of the error and the situation in which it occurred to TIBCO Support.
Errors	Error processing xa start request, %s. Error trying to enter standby for '%s', %s.

Category	Unrecognized option
Description	The server's command line contains an unrecognized option.
Resolution	Run the server with the -help option and compare it with the command line containing the unrecognized option.
Errors	Unrecognized option: '%s'.

Index

A

- admin
 - connect 168
 - password 249
 - user 164
- admin user 249
- anonymous
 - user and security 249

C

- compiling samples 328
- compression, message 65
- configuring
 - external directory for authentication 206
 - LDAP 206
- connect
 - admin 168
- customer support xx

D

- definitions of properties 35
- delete subscriber 172
- disabled security 248
- durable subscriber 5, 70, 176
- dynamic queues 32
- dynamic topics 32

E

- emsntsreg 246

- export
 - topic property 39
- export property 39
 - topic 171
- export topic property 171
- extensions
 - message 69

F

- failover and heartbeat 294
- files, sample 328
- flow control 51

G

- group 205

H

- heartbeat
 - failover and 294

I

- import
 - queue property 39
 - topic property 39
- import property 39
- inheritance
 - property 46

inheritance of property 32

J

JNDI connections 255

static queues 255

static topics 255

L

LDAP 206

M

MapMessage 58

definition 58

extension 69

maxbytes property 46

maxmsgs 37

message

compression 65

extensions 69

maximum size 58

message pool 124

N

no-acknowledge receipt 71

No-Acknowledgement Receipt Mode 71

O

overflowPolicy 37

P

password

admin 249

permission

secure property and 36

properties

queue 34

topic 34

property 69

definitions 35

export 39

import 39

inheritance 32, 46

maxbytes 46

Q

queue import property 39

queue properties 34

queue property list 34

queues

dynamic 32

static 32

temporary 33

R

reserve memory 124

round-robin queue (non-exclusive) 40

S

sample files 328

samples

compiling 328

secure property and permission 36

- security
 - and anonymous user 249
 - disabled 248
 - main configuration file 203
- shared storage 295
- static queues 32
 - JNDI connections 255
- static topics 32, 255
- subscriber 5, 211
 - delete 172
 - durable 5, 70, 176
- support, contacting xx

T

- tcp 169, 253, 330
- technical support xx
- temporary queues 33
- temporary topics 33
- topic export property 39
- topic import property 39
- topic property list 34
- topics
 - dynamic 32
 - static 32
 - temporary 33

U

- UNIX system
 - using for user authentication 206
- user 205
 - admin 249
 - externally authenticated 206
- user admin 164

TIBCO Software Inc. End User License Agreement

READ THIS END USER LICENSE AGREEMENT CAREFULLY. BY DOWNLOADING OR INSTALLING THE SOFTWARE, YOU AGREE TO BE BOUND BY THIS AGREEMENT. IF YOU DO NOT AGREE TO THESE TERMS, DO NOT DOWNLOAD OR INSTALL THE SOFTWARE AND RETURN IT TO THE VENDOR FROM WHICH IT WAS PURCHASED.

Upon your acceptance as indicated above, the following shall govern your use of the Software except to the extent all or any portion of the Software (a) is subject to a separate written agreement, or (b) is provided by a third party under the terms set forth in an Addenda at the end of this Agreement, in which case the terms of such addenda shall control over inconsistent terms with regard to such portion(s).

License Grant. The Software is the property of TIBCO or its licensors and is protected by copyright and other laws. While TIBCO continues to own the Software, TIBCO hereby grants to Customer a limited, non-transferable, non-exclusive, license to use the number of Permitted Instances set forth in the Ordering Document, in machine-readable, object code form and solely for Customer's internal business use.

Restrictions. Customer agrees not to (a) make more copies than the number of Permitted Instances plus a reasonable number of backups; (b) provide access to the Software to anyone other than employees, contractors, or consultants of Customer; (c) sublicense, transfer, assign, distribute to any third party, pledge, lease, rent, or commercially share the Software or any of Customer's rights under this Agreement (for the purposes of the foregoing a change in control of Licensee is deemed to be an assignment); (d) use the Software for purposes of providing a service bureau, including, without limitation, providing third-party hosting, or third-party application integration or application service provider-type services, or any similar services; (e) use the Software in connection with ultrahazardous activities, or any activity for which failure of the Software might result in death or serious bodily injury to Customer or a third party; or (f) directly or indirectly, in whole or in part, modify, translate, reverse engineer, decrypt, decompile, disassemble, make error corrections to, create derivative works based on, or otherwise attempt to discover the source code or underlying ideas or algorithms of the Software.

Beta and Evaluation Licenses. Notwithstanding the foregoing, if the Software is being provided for demonstration, beta testing, or evaluation purposes, then Customer agrees (a) to use the Software solely for such purposes, (b) that the Software will not be used or deployed in a production environment, and (c) that such use shall automatically terminate upon the earlier of thirty days from the date Customer receives the right to install the Software, or Customer's receipt of notice of termination from TIBCO.

Technical Support. Provided Customer has paid applicable support fees (not included with Software fees unless separately listed), TIBCO shall provide support for generally available TIBCO Software on an annual basis commencing on the Purchase Date, as follows ("Support"): Customer shall designate at TIBCO's support website <https://support.tibco.com/eSupport/newuser.html>, the number of technical support contacts permitted under the level of Support purchased (contacts are changeable upon 48-hours prior written notice to TIBCO). Each contact may contact TIBCO for problem resolution during TIBCO's published support hours corresponding to the level of Support fees paid.

Upon notice from a contact of a Software problem which can be reproduced at a TIBCO support facility or via remote access to

Customer's facility, TIBCO shall use reasonable efforts to correct or circumvent the problem according to its published support objectives. TIBCO reserves the right to make changes only to the most currently available version. TIBCO will use reasonable efforts to support the previously released version of the Software for a maximum of six months.

TIBCO shall have no obligation to support the Software (i) for use on any computer system running other than the operating system software for which the Software is approved (as set forth in the Software documentation) and licensed hereunder, or (ii) if Customer has modified or authorized a third party to modify the Software. TIBCO shall have no obligation to modify any version of the Software to run with any new versions of any operating system, or any other third party software or hardware. If Customer purchases Support for any Software, Customer must purchase the same level of Support for all copies of the Software for which it is licensed.

Support may be extended for one-year periods on the anniversary of each Purchase Date at the standard amounts set forth in its price list, for as long as TIBCO offers Support. Customer may reinstate lapsed support for any then currently supported Software by paying all Support fees in arrears and any applicable reinstatement fee. Upgrades, patches, enhancements, bug fixes, new versions and/or new releases of the Software provided from time to time under Support shall be used only as replacements to existing Permitted Instances, and shall not be deemed to increase that number, and use thereof shall be governed by the terms of this Agreement, except for the first paragraph of the Limited Warranty and any right of return or refund.

Consulting Services. Customer may request additional services ("Services") either in an Ordering Document, or by a separate mutually executed work order, statement of work or other work-request document incorporating this Agreement (each, a "Work Order"). Unless otherwise expressly agreed to in a Work Order, all Services and any work product therefrom shall be (a) performed on a time and materials basis, plus meals, lodging, travel, and other expenses reasonably incurred in connection therewith, (b) deemed accepted upon delivery, and (c) exclusively owned by TIBCO (except for confidential information of Customer identified to TIBCO in the Ordering Document), including all right, title and intellectual property or other right or interest therein. Each Work Order is intended to constitute an independent and distinct agreement of the parties, notwithstanding that each shall be construed to incorporate all applicable provisions of this Agreement. Specific to TIBCO training services, additional information regarding courses, registration, restrictions or limitation can be found at TIBCO's website at <http://www.tibco.com/services/education> under Education Programs. Fees for Services shall be due and payable in United States dollars net 30 from the date of TIBCO's invoice.

Limited Warranty. If Customer obtained the Software directly from TIBCO, then TIBCO warrants that for a period of thirty (30) days from the Purchase Date: (i) the media on which the Software is furnished will be free of defects in materials and workmanship under normal use; and (ii) the Software will substantially conform to its published specifications. This limited warranty extends only to the original Customer hereunder. Customer's sole and exclusive remedy and the entire liability of TIBCO and its licensors under this limited warranty will be, at TIBCO's option, repair, replacement, or refund of the Software and applicable Support fees, in which event this Agreement shall terminate upon payment thereof.

This warranty does not apply to any Software which (a) is licensed for beta, evaluation, testing or demonstration purposes for which TIBCO does not receive a license fee, (b) has been altered or modified, except by TIBCO, (c) has not been installed, operated, repaired, or maintained in accordance with instructions supplied by TIBCO, (d) has been subjected to abnormal physical or electrical stress, misuse, negligence, or accident, or (e) is used in violation of any other term of this Agreement. Customer agrees to pay TIBCO for any Support or Services provided by TIBCO related to a breach of the foregoing on a time, materials, travel, lodging and other reasonable expenses basis. If Customer obtained the Software from a TIBCO reseller or distributor, the terms of any warranty shall be as provided by such reseller or distributor, and TIBCO provides Customer no warranty with respect to such Software.

EXCEPT AS SPECIFIED IN THIS LIMITED WARRANTY, THE SOFTWARE, SUPPORT AND SERVICES ARE PROVIDED "AS IS", ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS, AND WARRANTIES INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTY OR CONDITION OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, SATISFACTORY QUALITY OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE, ARE HEREBY EXCLUDED TO THE EXTENT ALLOWED BY APPLICABLE LAW. NO WARRANTY IS MADE REGARDING THE RESULTS OF ANY SOFTWARE, SUPPORT OR SERVICES OR THAT THE SOFTWARE WILL OPERATE WITHOUT ERRORS, PROBLEMS OR INTERRUPTIONS, OR THAT ERRORS OR BUGS IN THE SOFTWARE WILL BE CORRECTED, OR THAT THE SOFTWARE'S FUNCTIONALITY OR SERVICES WILL MEET CUSTOMER'S REQUIREMENTS. NO TIBCO DEALER, DISTRIBUTOR, AGENT OR EMPLOYEE IS AUTHORIZED TO MAKE ANY MODIFICATIONS, EXTENSIONS OR ADDITIONS TO THIS WARRANTY.

Indemnity. If Customer obtained the Software from TIBCO directly, then TIBCO shall indemnify Licensee from and against any final judgment by a court of competent jurisdiction, including reasonable attorneys' fees, that the unmodified TIBCO Software infringes any patent issued by the United States, Canada, Australia, Japan, or any member of the European Union, or any copyright, or any trade secret of a third party; provided that TIBCO is promptly notified in writing of such claim, TIBCO has the exclusive right to control such defense and/or settlement, and Licensee shall provide reasonable assistance (at TIBCO's expense) in the defense thereof. In no event shall Licensee settle any claim, action or proceeding without TIBCO's prior written approval. In the event of any such claim, litigation or threat thereof, TIBCO, at its sole option and expense, shall (a) procure for Licensee the right to continue to use the TIBCO Software or (b) replace or modify the TIBCO Software with functionally equivalent software. If such settlement or modification is not commercially reasonable (in the reasonable opinion of TIBCO), TIBCO may cancel this Agreement upon sixty days prior written notice to Licensee, and refund to Licensee the unamortized portion of the license fees paid to TIBCO by Licensee based on a five-year straight-line depreciation. This Section states the entire liability of TIBCO with respect to the infringement of any Intellectual Property rights, and Licensee hereby expressly waives any other liabilities or obligations of TIBCO with respect thereto. The foregoing indemnity shall not apply to the extent any infringement could have been avoided by use of the then-current release.

Limitation of Liability. EXCEPT AS PROVIDED UNDER INDEMNITY OR RESULTING FROM A BREACH OF CONFIDENTIALITY (THE "EXCLUDED MATTERS"), IN NO EVENT WILL EITHER PARTY OR TIBCO'S LICENSORS BE LIABLE FOR ANY LOST DATA, LOST REVENUE, LOST PROFITS, DAMAGE TO REPUTATION, BUSINESS INTERRUPTION, OR ANY OTHER

INDIRECT, INCIDENTAL, CONSEQUENTIAL, SPECIAL, PUNITIVE, EXEMPLARY OR ANY SIMILAR TYPE DAMAGES ARISING OUT OF THIS AGREEMENT, THE USE OR THE INABILITY TO USE THE SOFTWARE, OR THE PROVISION OF ANY SUPPORT OR SERVICES, EVEN IF A PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. EXCEPT FOR THE EXCLUDED MATTERS, IN NO EVENT SHALL A PARTY BE LIABLE TO THE OTHER, WHETHER IN CONTRACT, TORT (INCLUDING ACTIVE OR PASSIVE NEGLIGENCE), BREACH OF WARRANTY, CLAIMS BY THIRD PARTIES OR OTHERWISE, EXCEED THE PRICE PAID BY CUSTOMER UNDER THE APPLICABLE ORDERING DOCUMENT.

THE FOREGOING LIMITATIONS SHALL APPLY EVEN IF THE ABOVE-STATED REMEDY OR LIMITED WARRANTY FAILS OF ITS ESSENTIAL PURPOSE. BECAUSE SOME STATES OR JURISDICTIONS DO NOT ALLOW LIMITATION OR EXCLUSION OF CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO CUSTOMER.

Confidentiality. "Confidential Information" means the terms of this Agreement; all information marked by the disclosing party as proprietary or confidential; any provided software, related documentation or related performance test results derived by Licensee; and any methods, concepts or processes utilized in provided software or related documentation. Confidential Information shall remain the sole property of the disclosing party and shall not be disclosed to any non-Authorized User without the prior written consent of the disclosing party. If Confidential Information is communicated orally, such communication shall be confirmed as "Confidential" in writing within thirty days of such disclosure. The parties agree to protect the Confidential Information of the other in the same manner it protects the confidentiality of similar information and data of its own (and at all times exercising at least a reasonable degree of care). Except with respect to the Software, items will not be deemed Confidential Information if (i) available to the public other than by a breach of an agreement with TIBCO, (ii) rightfully received from a third party not in breach of any obligation of confidentiality, (iii) independently developed by one party without use of the Confidential Information of the other; (iv) known to the recipient at the time of disclosure (other than under a separate confidentiality obligation); or (v) produced in compliance with applicable law or court order, provided the other party is given reasonable notice of the same. Both parties agree to indemnify the other for any damages the other may sustain resulting from their unauthorized use and/or disclosure of the other's Confidential Information. Such damages shall include reasonable expenses incurred in seeking both legal and equitable remedies. To the extent required by law, at Customer's request, TIBCO shall provide Customer with the interface information needed to achieve interoperability between the Software and another independently created program, on payment of TIBCO's applicable fee. Customer agrees to observe obligations of confidentiality with respect to such information.

Export. Software, including technical data, is subject to U.S. export control laws, including the U.S. Export Administration Act and its associated regulations, and may be subject to export or import regulations in other countries. Customer agrees to comply strictly with all such regulations and agrees to obtain all necessary licenses to export, re-export, or import Software.

Government Use. If the Customer is an agency, department, or other entity of the United States Government ("Government"), the use, duplication, reproduction, release, modification, disclosure or transfer of the Software, or any related documentation of any kind, including technical data or manuals, is restricted in accordance with Federal Acquisition Regulation ("FAR") 12.212 for civilian agencies and

Defense Federal Acquisition Regulation Supplement ("DFARS") 227.7202 for military agencies. The Software is commercial computer software and commercial computer software documentation. Use of the Software and related documentation by the Government is further restricted in accordance with the terms of this Agreement, and any modification thereto.

Orders. An Ordering Document shall be deemed accepted only by issuance of a TIBCO invoice and solely for purposes of administrative convenience. None of the terms of the Ordering Document (other than the Software product name, number of Permitted Instances, level of Support, description of Services, and fees due in connection therewith) shall apply for any reason or purpose whatsoever, regardless of any statement on any Ordering Document to the contrary, unless countersigned by an officer of TIBCO. This Agreement constitutes the entire agreement between the parties with respect to the use of the Software, Support and Services, and supersedes all proposals, oral or written, and all other representations, statements, negotiations and undertakings relating to the subject matter hereof. All orders of Software, Support or Services by Customer from TIBCO shall be deemed to occur under the terms of this Agreement (with or without reference to this Agreement), unless expressly superseded by a signed written Agreement between the parties. Software shall be delivered electronically, and such delivery shall occur when the TIBCO Software is made available for download by Customer. Physical deliveries (as applicable) of Software and documentation which typically accompanies the Software on delivery shall be on CD-ROM, FOB Palo Alto, and delivery shall occur by depositing the CD-ROM with TIBCO's overnight carrier (at no charge to Customer).

Term and Termination. Support or Services may be terminated: (a) by either party upon a default of the other, such default remaining uncured for fifteen days from written notice from the non-defaulting party; (b) upon the filing for bankruptcy or insolvency of the other party; (c) by either party upon prior written notice at least sixty days prior to the end of any annual Maintenance period; or (d) by Licensee (for Services), upon ten days prior written notice. Termination of Support or Services shall not terminate this Agreement. Customer may terminate this Agreement in its entirety at any time by destroying all copies of the Software. Upon termination of this Agreement in its entirety, for any reason, Customer must cease using and return or destroy all copies of the Software. Customer's obligation to pay accrued charges and any fees due as of the date of termination, as well as the sections entitled "Confidentiality", "Limited Warranty" and "Limitation of Liability" shall survive any such termination.

Authority. You hereby represent and warrant that you have full power and authority to accept the terms of this Agreement on behalf of Customer, and that Customer agrees to be bound by this Agreement.

General. Fees on the Ordering Document (all to be paid on the latter of thirty days from Invoice by TIBCO or the date set forth in the Ordering Document) do not include sales, use, withholding, value-added or similar taxes, and Customer agrees to pay the same, excluding therefrom taxes related to TIBCO's income and corporate franchise tax. Customer agrees to pay all reasonable costs incurred (including reasonable attorneys' fees) in collecting past due amounts under this Agreement. Except as set forth in the Section entitled "Warranty" all fees paid under or in connection with this Agreement are non-refundable and no right of set-off exists. All payments of fees due shall be made in U.S. dollars, net 30 from Purchase Date, or, for any other amounts coming due hereafter, net 30 from TIBCO's invoice. A service charge of one and one-half percent per month will be applied to all invoices that are not paid on time. Licensee agrees to pay all sales, use, value-added, withholding, excise and any other similar taxes or government charges, exclusive

of TIBCO's income taxes. No delay in the performance of any obligation by either party, excepting all obligations to make payment, shall constitute a breach of this Agreement to the extent caused by force majeure. Customer hereby grants TIBCO and its independent auditors the right to audit Customer's compliance with this Agreement. If any portion of this Agreement is found to be void or unenforceable, the remaining provisions shall remain in full force and effect. This Agreement shall be governed by and construed in accordance with the laws of the State of California, United States of America, as if performed wholly within the state and without giving effect to the principles of conflict of law. The state and/or federal courts in San Francisco, California, shall have exclusive jurisdiction of any action arising out of or relating to this Agreement. The United Nations Convention on Contracts for the International Sale of Goods is excluded from application hereto. If any portion hereof is found to be void or unenforceable, the remaining provisions of this Agreement shall remain in full force and effect.

Definitions. In connection with this Agreement, the following capitalized terms shall have the following meaning: "Agreement" means this End User License Agreement; "Case Start" means the initiation of a single instance of a defined business process; "Connection" for the following TIBCO Software products shall mean: for TIBCO Enterprise Message Service, a TIBCO Enterprise Message Service client connection to the TIBCO Enterprise Message Service server for the purpose of sending or receiving messages, for TIBCO SmartSockets and TIBCO SmartMQ, any network protocol link established with such TIBCO Software (directly or indirectly) to any other entity, including but not limited to software, firmware or hardware, for TIBCO Enterprise RTView - Standard Monitor System, the number of monitored server instances to TIBCO Rendezvous daemons or TIBCO Hawk agents; for TIBCO Enterprise RTView- EMS Monitor System, a monitored TIBCO Enterprise Message Service Connection (as defined above for that product); for TIBCO General Interface, an electronic data interface to a CPU on a server (which excludes CPUs on devices such as routers, switches, proxies, or HTTP or application servers configured to substantially pass-through information or messages to TIBCO General Interface) that produces information or messages consumed by TIBCO General Interface; "Customer" means the original purchaser or licensee of the Software and any permitted successors and assigns; "Developer" means one user/developer of a TIBCO Software product for use in Development; "Development" means used for software development purposes only; "Enterprise" means an unlimited number of Permitted Instances for a period of one year from the Purchase Date (unless otherwise set forth in the Ordering Document), at which time existing licenses convert to perpetual and Customer may not thereafter deploy additional Permitted Instances, and in any event, shall (during the one-year unlimited deployment period) exclude any entity which acquires, is acquired by, merged into, or otherwise combined with Customer. Customer hereby agrees to provide TIBCO with notice of the number of Permitted Instances deployed at the end of such one-year period within thirty days thereafter; "Fab" means unlimited use for shop-floor manufacturing applications at a Site; "Workstation" shall mean a single end-user computer that is generally intended to be accessed by one person at a time; "Ordering Document" means any purchase order or similar document or agreement requesting Software, Support or Services; "Permitted Instance(s)" means the number of copies of Software running on a Server Instance, Workstation, User, or Development basis, on a designated Platform, as set forth in an Ordering Document, including, without limitation, Enterprise, Site and Fab licensing; "Platform" means the operating system set forth in an Ordering Document; "Purchase Date" means the date of the Ordering Document is accepted by TIBCO; "Server Instance" means a computer with 1 CPU (unless otherwise set forth in the Ordering Document) performing common services for multiple machines; "Site" means an unlimited number of Permitted Instances at a specific

physical address set forth in the Ordering Document (or, in the absence of any address, at Customer's corporate headquarters); "Software" means the software products listed in an Ordering Document (except as provided in the second paragraph hereof), in whole and in part, along with their associated documentation; "TIBCO" means TIBCO Software Inc.; and "Named User" means the number of named users with access to the Software.

Special Product Provisions. TIBCO BusinessPartner: Customer may sublicense to third parties ("Partners") up to the total Number of Copies of TIBCO BusinessPartner, provided that for every such sublicense, the Number of Copies Customer is licensed to use shall be reduced by the same number, and provided further that prior to delivery of TIBCO BusinessPartner to a Partner, such Partner agrees in writing (a) to be bound by terms and conditions at least as protective of TIBCO as the terms of this Agreement, (b) that TIBCO BusinessPartner be used solely to communicate with Customer's implementation of TIBCO BusinessConnect, and (c) for such Partner to direct all technical support and Maintenance questions directly to Customer. Customer agrees to keep records of the Partners to which it distributes TIBCO BusinessPartner, and to provide TIBCO the names thereof (with an address and contact name) within sixty days of the end of each quarter. Third Party Software: Use of any other third-party software identified by its company and/or product name or otherwise designated in Licensee's Ordering Document (collectively "Third Party Software") is subject solely to the terms and conditions of the click-wrap or shrink-wrap license agreement included with the Third Party Software products, and for which TIBCO shall be an intended third-party beneficiary of same. TIBCO shall have no obligation whatsoever in connection with the Third Party Software (including, without limitation, any obligation to provide maintenance or support) and the provision of Third Party Software is accomplished solely as an accommodation and in lieu of Customer purchasing a license to Third Party Software directly from the third party vendor. Embedded/Bundled Products: Some TIBCO Software embeds or bundles other TIBCO Software (e.g., TIBCO InConcert bundles TIBCO Rendezvous). Use of such embedded or bundled TIBCO Software is solely to enable the functionality of the TIBCO Software licensed on the Cover Page, and may not be used or accessed by any other TIBCO Software, or for any other purpose. Open Source Software: If Licensee uses Open Source software in conjunction with the TIBCO Software, Licensee must ensure that its use does not: (i) create, or purport to create, obligations of use with respect to the TIBCO Software; or (ii) grant, or purport to grant, to any third party any rights to or immunities under TIBCO's intellectual property or proprietary rights in the TIBCO Software. You also may not combine the TIBCO Software with programs licensed under the GNU General Public License ("GPL") in any manner that could cause, or could be interpreted or asserted to cause, the TIBCO Software or any modifications thereto to become subject to the terms of the GPL.

TIBCO EULA version 5.2, 3/05

Third Party Software Notices

Java Secure Socket Extension (JSSE) 1.0.3

This product includes code licensed from RSA Security, Inc.

OpenLDAP 2.1.30

Copyright© 1999-2003 The OpenLDAP Foundation, Redwood City, California, USA. All Rights Reserved.

THIS SOFTWARE IS PROVIDED BY THE OPENLDAP FOUNDATION AND ITS CONTRIBUTORS "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OPENLDAP FOUNDATION, ITS CONTRIBUTORS, OR THE AUTHOR(S) OR OWNER(S) OF THE SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

OpenSSL 0.9.7i

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit.

(<http://openssl.org/>).

Copyright© 1998-2004 The Open SSL Project. All Rights Reserved.

ADDENDA: Third Party License Agreements

Third Party Software License Agreements

The following are the software licenses for the Third Party Software provided in connection with the software.

Open SSL v2/v3 0.9.7i

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of the OpenSSL License and the original SSLeay license apply to the toolkit. See below for the actual license texts. Actually both licenses are BSD-style Open Source licenses. In case of any license issues related to OpenSSL please contact openssl-core@openssl.org.

OpenSSL License

Copyright (c) 1998-2003 The OpenSSL Project. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (<http://www.openssl.org/>)"
4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org.
5. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project.
6. Redistributions of any form whatsoever must retain the following acknowledgment:

"This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product includes cryptographic software written by Eric Young (ey@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

Original SSLeay License

Copyright (C) 1995-1998 Eric Young (ey@cryptsoft.com)

All rights reserved.

This package is an SSL implementation written by Eric Young (ey@cryptsoft.com).

The implementation was written so as to conform with Netscape's SSL.

This library is free for commercial and non-commercial use as long as the following conditions are adhered to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, lhash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed.

If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used.

This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement: "This product includes cryptographic software written by Eric Young (ey@cryptsoft.com)" The word 'cryptographic' can be left out if the routines from the library being used are not cryptographic related:-).
4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement:

"This product includes software written by Tim Hudson (tjh@cryptsoft.com)"

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL

DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The licence and distribution terms for any publicly available version or derivative of this code cannot be changed, i.e. this code cannot simply be copied and put under another distribution licence [including the GNU Public Licence.]

OpenLDAP 2.1.30

The OpenLDAP Public License Version 2.8, 17 August 2003

Redistribution and use of this software and associated documentation ("Software"), with or without modification, are permitted provided that the following conditions are met:

1. Redistributions in source form must retain copyright statements and notices,
2. Redistributions in binary form must reproduce applicable copyright statements and notices, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution, and
3. Redistributions must contain a verbatim copy of this document.

The OpenLDAP Foundation may revise this license from time to time. Each revision is distinguished by a version number. You may use this Software under terms of this license revision or under the terms of any subsequent revision of the license.

THIS SOFTWARE IS PROVIDED BY THE OPENLDAP FOUNDATION AND ITS CONTRIBUTORS "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OPENLDAP FOUNDATION, ITS CONTRIBUTORS, OR THE AUTHOR(S) OR OWNER(S) OF THE SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The names of the authors and copyright holders must not be used in advertising or otherwise to promote the sale, use or other dealing in this Software without specific, written prior permission. Title to copyright in this Software shall at all times remain with copyright holders.

OpenLDAP is a registered trademark of the OpenLDAP Foundation.

Copyright 1999-2003 The OpenLDAP Foundation, Redwood City, California, USA. All Rights Reserved. Permission to copy and distribute verbatim copies of this document is granted.

X Open ICU 1.4.1.2 Source License

COPYRIGHT AND PERMISSION NOTICE

Copyright (c) 1999,2000,2001 Compaq Computer Corporation
 Copyright (c) 1999,2000,2001 Hewlett-Packard Company
 Copyright (c) 1999,2000,2001 IBM Corporation
 Copyright (c) 1999,2000,2001 Hummingbird Communications Ltd.
 Copyright (c) 1999,2000,2001 Silicon Graphics, Inc.
 Copyright (c) 1999,2000,2001 Sun Microsystems, Inc.
 Copyright (c) 1999,2000,2001 The Open Group
 All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

X Window System is a trademark of The Open Group.

OSF/1, OSF/Motif and Motif are registered trademarks, and OSF, the OSF logo, LBX, X Window System, and Xinerama are trademarks of the Open Group. All other trademarks and registered trademarks mentioned herein are the property of their respective owners.

zlib 1.2.3

This product includes zlib software, copyright 1995–2003 Jean-loup Gailly and Mark Adler

Apache Xalan-Java 2.6.0 and Apache Xerces for Java 2.6.2

The Apache Software License, Version 1.1

Copyright (c) 2000 The Apache Software Foundation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).". Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Apache" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.

5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foundation, please see <http://www.apache.org/>.

Portions of this software are based upon public domain software originally written at the National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign.

libxml 2.6.9

Copyright (C) 1998-2002 Daniel Veillard. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE DANIEL VEILLARD BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF

CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of Daniel Veillard shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from him.

