SUN SEEBEYOND

eGATE™ INTEGRATOR JMS REFERENCE GUIDE

Release 5.1.3

Sun microsystems

Part Number: 820-0948-10

Version 20070501182633

Contents

Chapter 8

Chapter 9

List of Figures

List of Tables

Chapter 1

Introduction

This *Introduction* describes the scope and organization of this document, and provides references to additional sources of relevant information.

What's in This Chapter

1.1 About This JMS Reference

This section provides information about this document, including an overview of its contents, scope, and intended audience.

1.1.1 What's in This JMS Reference

This Reference covers the following topics.

1. **Introduction** on page 10 describes the purpose of this document, including writing conventions and a list of related documents.
2. **Java Message Service and Java CAPS** on page 13 summarizes how JMS is implemented in Java CAPS.
3. **Inside the Sun SeeBeyond JMS Provider** on page 18 provides detailed information about how the JMS IQ Manager processes messages.
4. **The JMS OTD** on page 37 provides detailed information about the JMS OTD.
5. **JMS OTD Methods** on page 43 describes JMS OTD methods and properties used in building Java Collaboration Definitions.
6. **JMS Message Methods** on page 65 describes JMS methods and properties that are used with JMS messages in Java Collaboration Definitions.
7. **JMS Client Configuration** on page 85 describes JMS client configuration properties.

8 **JMS IQ Manager Runtime Configuration** on page 94 describes JMS IQ Manager configuration properties.

9 **JMS IQ Manager Management** on page 110 describes how use the MS Control utility to manage JMS IQ Managers.

10 **Additional Java CAPS Message Servers** on page 121 provides information on additional JMS message servers (other than the JMS IQ Manager) supported by Enterprise Designer.

11 **Troubleshooting** on page 125 contains tips on troubleshooting problems in the JMS IQ Manager.

In addition, the **Glossary** on page 126 lists various terms used in this Reference.

1.1.2 Scope

This JMS Reference describes the Java™ Message Service as implemented in the Sun Java™ Composite Application Platform Suite (Java CAPS). It also describes how to configure and manage the various message servers supported by Java CAPS.

This Reference refers extensively to the *Sun SeeBeyond eGate™ Integrator User's Guide* for Enterprise Designer-specific procedures, and to the *Sun SeeBeyond eGate™ Integrator System Administration Guide* for Enterprise Manager-specific procedures.

1.1.3 Intended Audience

The JMS Reference is intended for use by Java CAPS Project designers and administrators having detailed knowledge of the JMS API and the *Sun Java™ Message Server Specification* version 1.1.

1.1.4 Text Conventions

The following conventions are observed throughout this document.

Table 1 Text Conventions

Text	Used For	Example
Bold	Names of buttons, files, icons, parameters, variables, methods, menus, and objects.	▪ Click **OK** to save and close. ▪ From the **File** menu, select **Exit**. ▪ Select the **logicalhost.exe** file. ▪ Enter the **timeout** value. ▪ Use the **getClassName()** method. ▪ Configure the **Inbound** File eWay.
`Monospaced`	Command line arguments, code samples (variables are shown in ***bold italic***).	`bootstrap -p` ***`password`***
Blue bold	Hypertext links within document.	See **Text Conventions** on page 11

Table 1 Text Conventions (Continued)

Text	Used For	Example
Blue underlined	Hypertext links for Web addresses (URLs) or email addresses.	http://www.sun.com CAPS_docsfeedback@sun.com

1.1.5 Screenshots

Depending on what products you have installed, and how they are configured, the screenshots in this document may differ from what you see on your system.

1.2 Related Documents

The following documents provide additional information about Java CAPS message servers:

- *Sun Java™ Composite Application Platform Suite Deployment Guide*
- *Sun Java™ Composite Application Platform Suite Installation Guide*
- *Sun Java™ Message Server Specification version 1.1*
- *Sun Java™ System Message Queue 3 Administration Guide*
- *Sun Java™ System Message Queue 3 Technical Overview*
- *Sun SeeBeyond eGate™ Integrator System Administration Guide*
- *Sun SeeBeyond eGate™ Integrator User's Guide*

1.3 Sun Microsystems, Inc. Web Site

The Sun Microsystems web site is your best source for up-to-the-minute product news and technical support information. The site's URL is:

http://www.sun.com

1.4 Documentation Feedback

We appreciate your feedback. Please send any comments or suggestions regarding this document to:

CAPS_docsfeedback@sun.com

Chapter 2

Java Message Service and Java CAPS

Java Message Service and Java CAPS summarizes how the Java Message Service (JMS™) is implemented and used in Java CAPS.

What's in This Chapter

2.1 Java Message Service

The Java Message Service is a Java™ API used for sending and receiving messages. It is vendor-agnostic, and is used almost universally in enterprise messaging systems such as that included in Java CAPS. JMS provides a standard, currently embodied in the JMS version 1.1 specification, which is an integral part of the Java 2, Enterprise Edition (J2EE) 1.4 platform.

The use of JMS allows loosely coupled, reliable, asynchronous interactions among J2EE components and legacy systems capable of messaging. Version 1.1 of the JMS API includes the following features:

- Message-driven beans, which enable the asynchronous consumption of JMS messages.
- Message sends and receives that can participate in Java Transaction API (JTA) transactions.
- J2EE Connector Architecture (JCA) interfaces that allow JMS implementations from different vendors to be externally plugged into a J2EE 1.4 application server.

Additionally, the J2EE platform's Enterprise JavaBeans (EJB) container architecture provides the following enhancements the JMS API:

- Allows the concurrent consumption of messages.

- Provides support for distributed transactions, so that database updates, message processing, and connections to EIS systems using the J2EE Connector Architecture can all participate in the same transaction context.

The features of the JMS version 1.1 specification have been widely adopted in the Sun SeeBeyond JMS implementation, and are described in this Reference Guide.

2.2 JMS Message Servers

JMS message servers provide the global messaging protocols, such as the routing and delivery of messages, and connection to the JMS database. By default, Enterprise Designer offers three built-in message server options:

- Sun SeeBeyond JMS IQ Manager
- Sun Java™ System Message Queue
- IBM WebSphere MQ (version 6.0 only)

Sun SeeBeyond JMS IQ Manager is eGate Integrator's native JMS message server implementation. The JMS IQ Manager conforms to the Java Message specification 1.1 and supports both topic (publish-and-subscribe) and queue (point-to-point) messaging styles. **Inside the Sun SeeBeyond JMS Provider** on page 18 includes information on how the JMS IQ Manager processes messages, in concert with the JMS clients. **JMS IQ Manager Runtime Configuration** on page 94 provides detailed information about the JMS IQ Manager property options.

Information regarding the Sun Java System Message Queues and the IBM WebSphere MQ is found in the documentation supplied with those products. For the Sun Java System Message Queues, refer also to **Using Sun Java System Message Queues** on page 122.

Another option, which must be installed separately, is the Sun Java Message Service Grid (JMS Grid). Procedures for installing and configuring JMS Grid for use with Enterprise Designer are summarized in **Using Sun JMS Grid Clients** on page 123. For information on JMS Grid itself, see the *Sun Java™ Message Service Grid User's Guide*.

To support legacy SeeBeyond Projects, eGate Integrator also includes the Schema Run-time Environment (SRE) JMS IQ Manager, which must be installed separately. See **Using SRE JMS IQ Managers** on page 121 for information.

Additionally, eGate Integrator supports selected third-party application servers, which contain their own message servers (see the *Sun SeeBeyond eGate™ Integrator System Administration Guide* for details).

2.3 JMS Message Destinations

A message destination is a container for stored data, and can follow either the JMS topic or queue model.

- A *topic* is a message destination that conforms to the publish-and-subscribe messaging paradigm.
- A *queue* is a message destination that conforms to the point-to-point messaging paradigm.

Each message destination has at least two JMS Clients associated with it: a *producer* client at its input, and a *consumer* client at (each) output. JMS message destinations are discussed in the *Projects* chapter of the *Sun SeeBeyond eGate™ Integrator User's Guide*.

2.4 JMS Clients

JMS clients provide the local messaging protocols, such as message persistence and delivery semantics, for messages being propagated between Project components. Together with the JMS message server, they constitute a *JMS provider*.

JMS clients are of two basic types: *producers* and *consumers* (or a combination of both). If associated with a queue, these become queue *senders* and *receivers*, respectively. If associated with a topic, they become topic *publishers* and *subscribers*, respectively.

JMS client configuration is discussed in the *Projects* chapter of the *Sun SeeBeyond eGate™ Integrator User's Guide*, with additional information in **JMS Client Configuration** on page 85 of this JMS Reference.

2.5 JMS OTDs

The Java Message Service (JMS) OTD acts as a "wrapper" around a message or connection, allowing Collaborations to read from and write to topics or queues. It indicates to the Collaboration which topic or queue it expects to receive messages from or send messages to, and allows you to build the JMS business rules. JMS OTD methods and properties are discussed in **JMS OTD Methods** on page 43. The JMS OTD is described further in the *Sun SeeBeyond eGate™ Integrator User's Guide*.

2.6 JMS Library File

JMS methods are contained in the JMS library file. In Java CAPS 5.1.x, this file is located in the following path (note that this is different from ICAN 5.0.x):

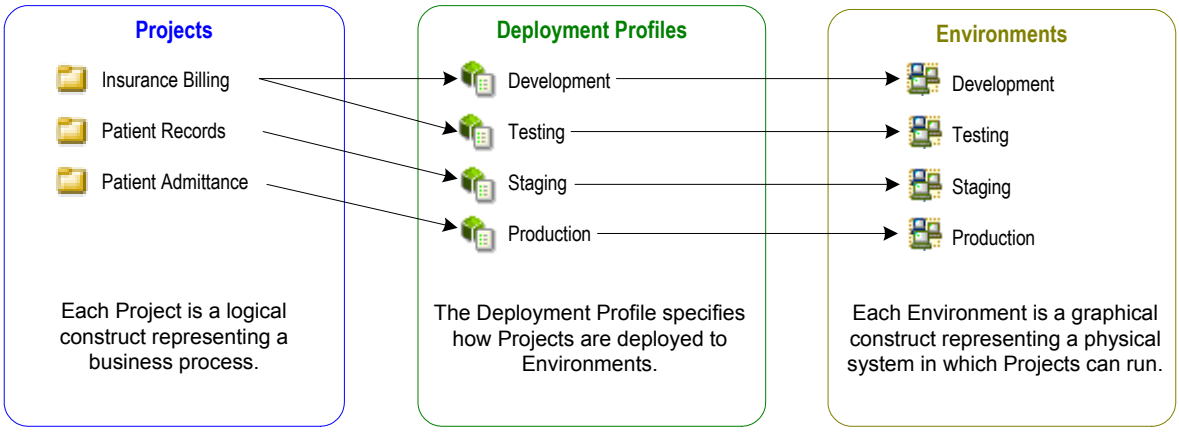
…\logicalhost\is\lib**com.stc.jms.stcjms.jar**

2.7 Implementing JMS in Java CAPS Projects

2.7.1 Integration Model

As described in the *Sun SeeBeyond eGate™ Integrator User's Guide*, Figure 1, below, illustrates the integration model used by Java CAPS.

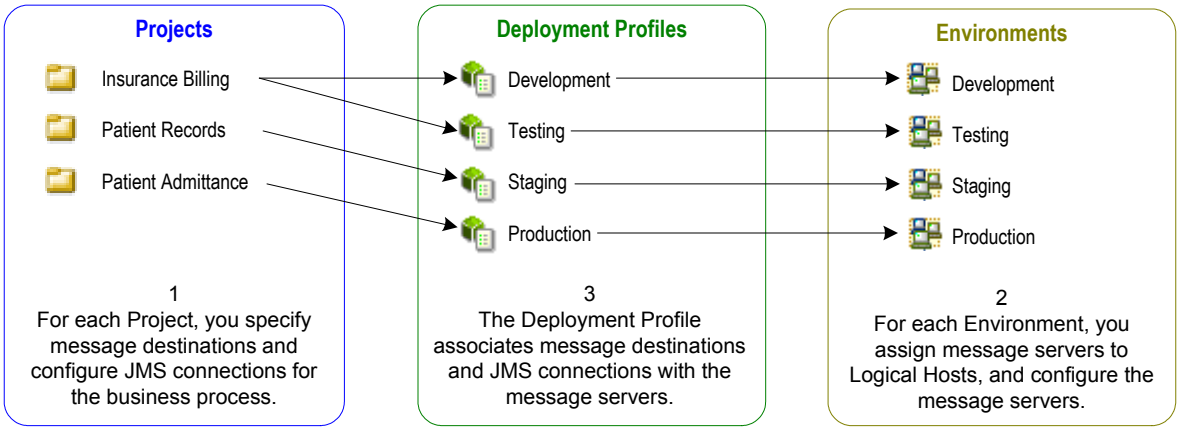
Figure 1 eGate Integration Model



In Figure 1, any of the Projects can be deployed to any of the Environments via the mapping defined in the Deployment Profiles. The example in the figure shows that the patient admittance Project is already in the production phase and therefore was deployed using the *production* Deployment Profile. The patient records Project is in the staging phase and was therefore deployed to the staging Environment using the *staging* Deployment Profile. The insurance billing Project is still being developed and tested, and therefore it is deployed to development and testing via the *development* and *testing* profiles.

Figure 2 illustrates how you go about implementing JMS following this model.

Figure 2 Implementing JMS in the eGate Integration Model



2.7.2 Creating and Configuring Message Destinations

In a Java CAPS Project, you specify the business logic for the eGate implementation. For each of the Project components you specify logical properties; these properties are independent from the physical implementation. For JMS-related components, Projects are where you add and name message destinations, by dragging and dropping topics and queue icons onto the Connectivity Map canvas (see the *Sun SeeBeyond eGate™ Integrator User's Guide* for information).

2.7.3 Creating OTDs and Collaborations

After having added the message destinations, you then create the Object Type Definitions (OTDs) and Collaboration Definitions. For any Java-based Collaboration Definition that reads from or writes to a JMS message destination, you must add the JMS web service. For information on the Java methods used in these Collaboration Definitions, see **JMS OTD Methods** on page 43 and **JMS Message Methods** on page 65.

2.7.4 Configuring JMS Clients

On links between message destinations and their subscribers and publishers, there is a JMS Client icon. By double-clicking the icon in the Connectivity Map, you can configure local connection properties such as persistent or non-persistent delivery mode, XA, and concurrent processing. Procedures for building your Projects are described in the *Sun SeeBeyond eGate™ Integrator User's Guide* and the *Sun SeeBeyond eGate™ Integrator Tutorial*. Details of the configuration properties for JMS Clients are given in **JMS Client Configuration** on page 85 of this JMS Reference.

2.7.5 Configuring Message Servers

In the run-time Environment, you specify which message servers are used, and which Logical Hosts (domains) they are to run on. Once you add a message server to a domain, you specify the physical configurations for the message server. You can configure global properties for JMS messaging such as the port number, message delivery order, tuning configurations, journaling options, and diagnostic options. Procedures for defining your Environment are described in the *Sun SeeBeyond eGate™ Integrator User's Guide*. Details of the configuration properties for the JMS IQ Manager are given in **JMS IQ Manager Runtime Configuration** on page 94.

2.7.6 Creating Component Mappings

When you define a Deployment Profile, you create mappings between Projects and Environments. In the Deployment Profile, you specify which components of the business process are located on which systems in a specific Environment. For the JMS, you specify which message destinations run on a particular message server. Note that inbound and outbound message destinations must be deployed to the same server. For more details, see *Sun SeeBeyond eGate™ Integrator User's Guide* and the *Sun SeeBeyond eGate™ Integrator Tutorial*.

Chapter 3

Inside the Sun SeeBeyond JMS Provider

Inside the Sun SeeBeyond JMS Provider describes the functional behavior of the Sun SeeBeyond JMS Provider, consisting of the Sun SeeBeyond JMS IQ Manager and JMS clients. It addresses topics such as the JMS IQ Manager database, message processing, performance, and optimization.

What's in This Chapter

3.1 JMS Messaging Features

This section provides a brief overview of messaging features offered by the JMS IQ Manager working together with the JMS clients, and includes the following topics:

3.1.1 Message Delivery Order

The JMS IQ Manager provides the following special facilities to maintain message order in concurrent processing and across message destinations.

Note: *These facilities are not mandated by the Java Message Server specification.*

- Configuring the JMS IQ Manager for special first-in, first-out (FIFO) ordering modes for queues.
- Specifying a set of message destinations (a time order group) for which fully serialized processing occurs.
- Configuring topics and queues for concurrent or serial processing.

The following sections describe each method of processing order. For detailed information about processing order, refer to **Message Processing Order** on page 22.

3.1.2 Message Producer Priorities

eGate Integrator enables you to set message priorities for topic publishers and queue senders. The priority level causes all messages produced by the client to have that same priority level. For more information, refer to **Message Producer Priorities** on page 27.

3.1.3 Distributed Transactions

The JMS IQ Manager enables you to configure JMS Client properties for distributed transactions using the XA protocol. For more information, refer to **Transaction Mode** on page 93.

3.1.4 Security

The eGate Integrator provides role-based security for the JMS IQ Manager by using authentication via file realm and LDAP. When authentication is enabled, access to the JMS IQ Manager is only granted when the connection has a valid user ID and password.

JMS IQ Manager security is disabled by default. To enable security, refer to the *Sun SeeBeyond eGate™ Integrator System Administration Guide*.

3.1.5 Run-Time Management

eGate Integrator provides two alternatives for managing JMS IQ Managers:

- **Enterprise Manager**

  Enterprise Manager offers a user-friendly, graphical interface containing comprehensive run-time management functions. Here, for example, you can monitor message destinations, and view message properties and payloads. For details, refer to the *Sun SeeBeyond eGate™ Integrator System Administration Guide*.

- **MS Control utility**

  The MS Control utility is a command-line utility that enables you to manage many advanced aspects of the JMS IQ Managers. For details, refer to **JMS IQ Manager Management** on page 110.

Note: *Other message servers supported by Java CAPS have their own run-time management tools.*

3.2 JMS IQ Manager Database

The JMS IQ Manager uses the JMS IQ Manager database to store persistent messages. The database is also used to store messages that are larger than can be kept in the JMS IQ Manager memory, which is determined by the cache size setting (by default 0.5 MB for Windows and 1 MB for UNIX). By default, JMS clients are configured for persistent messaging; therefore, in a default configuration, the database is used to store these messages. The messages are stored until they are consumed or until the duration set for the maximum time to live for a live message expires, which is 30 days by default. Therefore these messages are sometimes referred to as "live" messages.

3.2.1 Database Files

The database resides in the message server folder on the Logical Host. The database consists of a number of database files called *segments*. A segment is a diskspace store that is memory-mapped on the server. The segments act together to form the equivalent of a sequential database. By default, these files are named **stcms*.dbs**.

The JMS IQ Manager creates four segments in the database when it starts up initially. The default size of a segment is 8 MB on Windows and 16 MB on UNIX. The JMS IQ Manager creates as many segments as necessary. Before running your Java CAPS Project, it is important to set the segment size to a larger value than the largest transaction the JMS IQ Manager may need to process. *The JMS IQ Manager cannot accommodate a transaction that is larger than the segment size*.

Note: *The transaction size is the sum of the sizes of the individual messages involved.*

Configurations such as the database filenames, segments size, the maximum and minimum number of segments created can all be specified. For information, refer to **Segment Properties** on page 96.

3.2.2 Database Location

The JMS IQ Manager database resides in the directory specified by the JMS IQ Manager **Data directory** property as described in **Data Directory** on page 96. The standard location for the database files is as follows:

..\logicalhost\is\domains*domain_name*\stcms\instance1\stcms*.dbs

If journaling is enabled, the **instance1** directory also contains a **Journal** directory, unless another location has been specified for the **Journal Directory** property. The Journaling directory holds the journaling database files. For information, refer to **Journal Directory** on page 101. Journaling is disabled by default.

3.2.3 Database Configuration and Operation

Default Configuration

The default configuration for the JMS IQ Manager database is as follows:

- The database resides in the following directory:

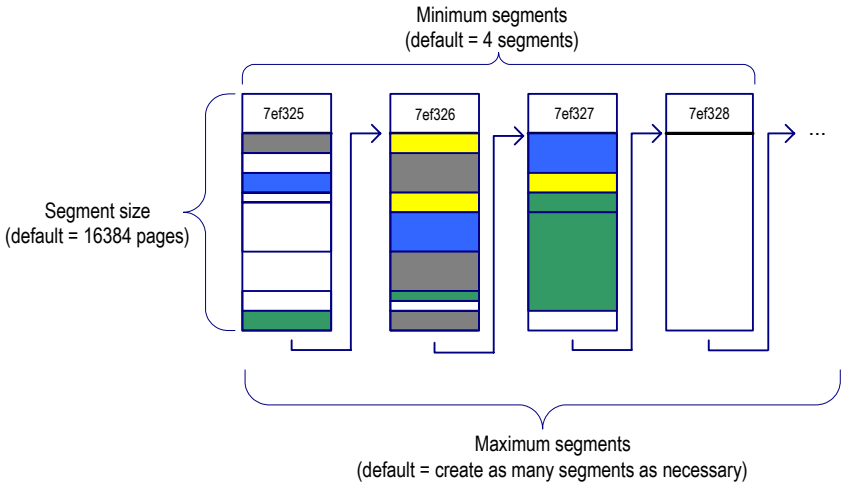
  ..\logicalhost\is\domains\domain_1\stcms\instance_1

- The segment filenames are **stcms*.dbs**.
- The number of segments created initially for the database is **4**.
- The size of each segment is **8** MB on Windows and **16** MB on UNIX.
- There is no limit to the number of segments that can be created.

This means that when the JMS IQ Manager starts up, the database consists of four segments; the total size of the database is **32** MB on Windows and **64** MB on UNIX.

Figure 3 shows a sample JMS IQ Manager database:

Figure 3 JMS IQ Manager Database Structure



Initial Operation

On startup, the JMS IQ Manager performs the following operations:

1 It allocates sufficient disk space to hold the minimum number of segments.

  Figure 3 shows a JMS IQ Manager allocation of four segments, numbered **7ef325** through **7ef328**.

2 As messages arrive, they are appended to the first segment until the segment is full.

  When a segment is full, the JMS IQ Manager stores subsequent messages in the first free segment.

  Figure 3 shows that the third segment, **7ef327** in file **stcms7ef327.dbs**, is almost full.

3 If there is no free segment, the JMS IQ Manager allocates a new segment if possible.

4 Memory is freed up as soon as the consumer has acknowledged the message or committed the transaction.

5 When all messages in a segment have expired or been removed from their queues, the JMS IQ Manager cleans up the segment, freeing it for re-use.

In Figure 3, the first segment (**7ef325**) has several segments that are white, indicating the slot is marked eligible. The segment is therefore almost ready for cleanup.

3.3 Message Processing Order

There are several ways to control the message processing order in Java CAPS Projects:

- You can specify first-in, first out (FIFO) ordering modes, or a set of message destinations with a specific processing order, when you configure the JMS IQ Manager (see **JMS IQ Manager Delivery Modes** on page 22).
- You can specify message processing (connection consumer or serial mode) at the JMS client level (see **JMS Client Delivery Modes** on page 25).

3.3.1 JMS IQ Manager Delivery Modes

For a single consumer with a single process, processing for queues is fully serialized, by default. The process of processing a message is as follows:

1 The receiver requests/is ready to receive a message

2 The receiver receives the message

3 The receiver processes the message

When multiple receivers or multiple processes within a single receiver subscribe to the same message destination (queues only), you have a choice of three first-in, first-out (FIFO) delivery modes, as listed below. These ordering modes apply globally to all queues in the Java CAPS Project

- **Fully concurrent**
  Receivers can retrieve messages when all older messages have been received, or are being received, and can commit messages in any order (without using time sequence).
- **Protected concurrent**
  Receivers can retrieve messages when all older messages have been received or are being received, but must commit using time sequence.
- **Fully serialized**
  Receivers read a messages only after all messages have been received and commit messages using time sequence.

Fully Concurrent Processing

In fully concurrent mode, receivers can retrieve messages from a destination only when all older messages have been received or are in the process of being received. Receivers can then commit messages without restrictions. By default, JMS IQ Managers use fully concurrent processing for queues.

Figure 4 Fully Concurrent Processing

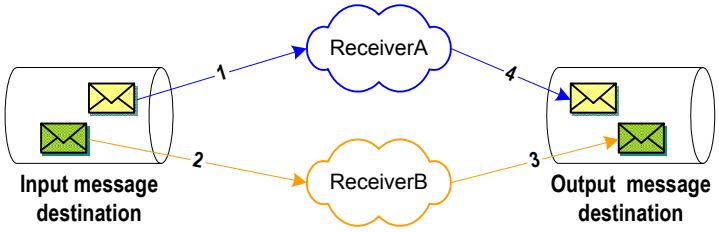


Figure 4 shows a sample delivery sequence for fully concurrent processing. In steps 1 and 2, the receivers retrieve their messages from the input queue. Both receivers must wait until each consumer has retrieved its messages (or is in the process of retrieving them) before they are able to commit messages to the output destination.

As steps 3 and 4 infer, receivers can commit messages in any order; therefore, messages can be committed out of sequence — which may or may not be acceptable. For example, a patient release record may be committed before the patient admittance record is committed. The table below indicates the benefits and costs of fully concurrent processing.

Table 2 Benefits and Costs — Fully Concurrent Processing

Benefits	Costs
Provides the highest performance.	Delivery not time-sequenced.
Receivers are not hampered by other receivers.	

Protected Concurrent Processing

In protected concurrent mode, receivers retrieve messages just as in fully concurrent mode (after all messages have been received or are being received), but messages can only be committed if all older messages have been committed.

Figure 5 Protected Concurrent Processing

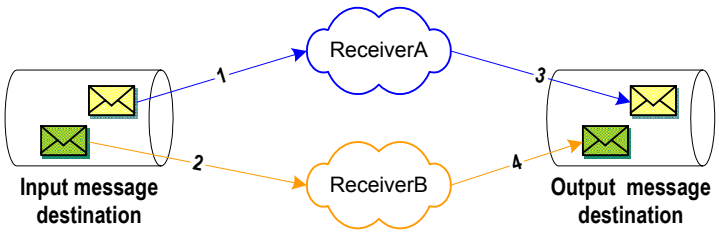


Figure 5 shows a sample delivery sequence for protected concurrent processing. In steps 1 and 2, the receivers retrieve their messages from the input queue. Both receivers must wait until each consumer has retrieved its messages (or is in the process of retrieving) before being able to commit messages to the output destination. ReceiverB might be ready to commit its message before ReceiverA, but must wait until ReceiverA commits its message (step 3). Only when ReceiverA's message has been committed, can ReceiverB commit its message (step 4).

Protected concurrent processing thus is a more workable solution in a scenario where a Project deals with messages such as patient records, where the admittance record must be committed before the release record.

The table below shows the benefits and costs of protected concurrent processing.

Table 3 Benefits and Costs — Protected Concurrent Processing

Benefits	Costs
Provides better performance than serialized processing.	Provides lower performance than fully concurrent processing.
Messages are delivered by time sequence.	

You specify protected concurrent processing for JMS IQ Managers with the **Protected Concurrent Queues** property as described in **Special FIFO Mode Properties** on page 103.

Note: *By design, protected concurrent FIFO works only within one transaction, either in the form of one session or in the form of one XA transaction on the same server. Using a single session in Java CAPS using an MDB is not possible, hence you should use XA when using protected concurrent FIFO with an MDB.*

Fully Serialized Processing

In fully serialized mode, receivers can only retrieve messages after all older messages have been received *and* committed.

Figure 6 Fully Serialized Processing

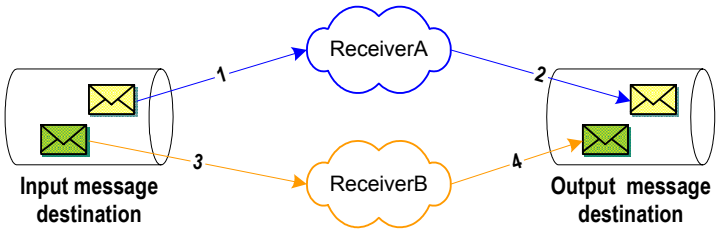


Figure 6 shows a sample delivery sequence for serialized processing. In step 1, ReceiverA retrieves its message. ReceiverB might at this point be ready to receive its message, but must wait until ReceiverA has committed its message. After ReceiverA commits the message in step 2, ReceiverB can then retrieve and commit its message (steps 3 and 4).

Table 4 shows the benefits and costs of protected concurrent processing.

Table 4 Benefits and Costs — Fully Serialized Processing

Benefits	Costs
Guaranteed delivery by time sequence.	Provides the lowest performance of all FIFO modes.

You specify fully serialized processing for JMS IQ Managers with the **Fully Serialized Queues** property as described in **Special FIFO Mode Properties** on page 103.

Serial Processing Across a Destination Group

You can also specify delivery order specifically for a set of topics and queues (time order groups). For these groups, consumers can only receive messages when all other messages in the time order group have been received or are in the process of being received. For information on specifying a time order group, refer to **Time Dependency Properties** on page 105.

3.3.2 JMS Client Delivery Modes

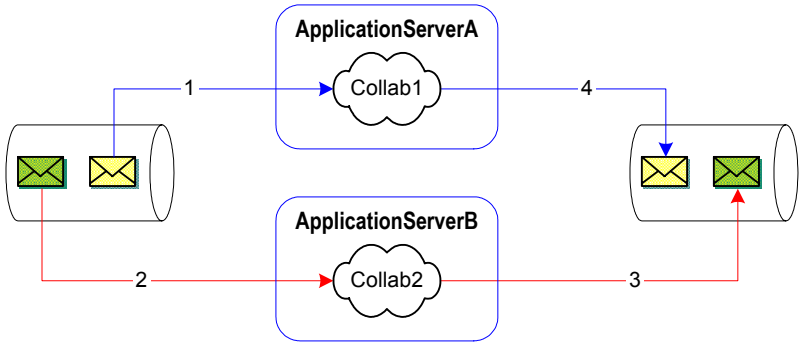
The delivery order options above are configured for the JMS IQ Manager. The eGate Integrator JMS implementation enables you to configure topic subscribers as connection consumers to improve message throughput through concurrent processing. You can set the JMS client configuration with the **Concurrency** property as described in **Concurrency** on page 87.

The use of connection consumers increases message processing performance by enabling concurrent processing via multiple threads. You can specify the number of message driven beans (MDBs) or server session pool to assign to a JMS Collaboration to process messages concurrently. When you use connection consumer with fully concurrent or protected concurrent FIFO processing, this default setting allows the integration server to assign multiple threads to execute the Collaboration on a particular message destination.

For queues, it is also possible to process messages concurrently using multiple integration servers; these integration servers may run on different systems.

Using a JMS client connection consumer affects the message processing order. For example, consider the scenario shown in Figure 7. The JMS IQ Manager is set to fully concurrent FIFO processing. However, each Collaboration on each integration server retrieves messages as they come in, and is able to commit them unrestricted to the queue. Therefore, although the JMS IQ Manager is configured for fully concurrent FIFO processing, message order cannot be guaranteed.

Figure 7 Multiple Application Server Configuration



The concurrency mode affects FIFO processing in several ways. Table 5 lists how the JMS client concurrency mode settings affect the JMS IQ Manager FIFO mode selections for topics. For topics, only one integration server per subscriber can be used.

Table 5 Concurrency/FIFO Mode Interaction — Topics

Concurrency Mode	Fully Concurrent FIFO Mode
serial mode	fully serialized
connection consumer	no strict order maintained

Table 6 lists how the JMS client concurrency mode settings affect the JMS IQ Manager FIFO mode selections for queues.

Table 6 Concurrency/FIFO Mode Interaction — Queues

Integration Servers	Concurrency Mode	FIFO Mode		
		Fully Serialized	Protected Concurrent	Fully Concurrent
single	serial	fully serialized	protected concurrent	fully serialized
		no concurrency	no concurrency	no concurrency
	connection consumer	fully serialized	protected concurrent	no strict order
		no concurrency	concurrent	concurrent
multiple	serial	fully serialized	protected concurrent	no strict order
		no concurrency	concurrent	concurrent
	connection consumer	fully serialized	protected concurrent	no strict order
		no concurrency	concurrent	concurrent

3.4 Message Producer Priorities

eGate Integrator enables you to set message priorities for topic publishers and queue senders. The priority is specified at the JMS client level; therefore, the level you specify causes all messages produced by that client to have that same priority level (unless overridden for a specific Collaboration — see below). For example, if you set the priority level to 2, all messages sent by that client have message priority level 2. The default message priority is 4.

The eGate message priority implementation adheres to the recommended standards in the Java Specification: in most circumstances, messages with higher priorities are delivered before message with lower priorities.

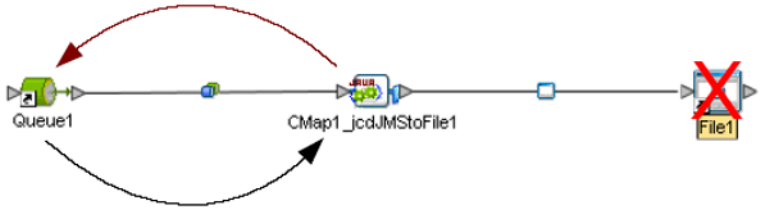
For information about setting the priority level for topic publishers and queue senders, refer to **Priority** on page 92.

You can also specify message priorities in specific Collaborations with the Collaboration Definition Editor. Collaboration message priorities override message priorities specified at the JMS client level. For more information, refer to the *Sun SeeBeyond eGate™ Integrator User's Guide*.

3.5 Message Redelivery and Redirection

JMS *receive* methods provide for the redelivery of messages that are rolled back when the message cannot be delivered (see Figure 8). JMSJCA (an implementation of the Java Connector Architecture 1.5) provides additional flexibility in the way messages are redelivered, and adds the option of redirecting the messages to another destination. The JMS IQ Manager currently supports these JMSJCA features.

Figure 8 Message Rollback and Redelivery



The reason for a message to be rolled back can be either permanent or transient in nature. If it is transient, the message will eventually be delivered after some number of retries; but if the number of retry/rollback cycles required to redeliver the message becomes large, system resources can be negatively impacted. To avoid this situation, a series of progressive delays, following a prescribed number of failed retries, is introduced into the redelivery process. The default behavior is described in Table 7.

Table 7 Default Redelivery Characteristics

Total Number of Failed retries	Delay (milliseconds)
3	25
5	50
10	100
20	1000
50	5000

You can override this default behavior by configuring your own custom characteristics. You specify the actions you want to be taken after a message has been rolled back by means of a specially-formatted string, which you append to the message server URL when you configure the JMS IQ Manager. The formatting of this string is best described by example, as presented in the following sections.

3.5.1 Redelivery Options

Progressive Delay

The format of such a delay is **<retries>:<delay>**, where the number of retries is counted from the original rollback and the delay time is given in milliseconds. The maximum allowed delay is five seconds (5000 ms). Example 1 shows a string that specifies a delay of 1000 milliseconds following 5 failed retries. In other words, it specifies no delay for the first 5 attempts at redelivery, then a 1 second delay for each subsequent attempt.

Example 1

```
5:1000
```

You can also cascade these delay actions to become progressively longer as the number of retries increases. Entries are separated by a semicolon followed by a space. Example 2 shows a string that specifies a 1 second delay following 5 failed attempts at redelivery, ana a 5 second delay following a total of 10 failed attempts.

Example 2

```
5:1000; 10:5000
```

Delay and Redirect

After a certain number of failed redelivery attempts, you may want to redirect the message to a different target destination, such as a dead-letter queue. The format for redirecting is **<retries>:move(args)**, where the arguments can be **queue:<target>**, **topic:<target>**, or **same:<target>**.

The argument component **same** specifies the same kind of message destination as the message source. That is, if the message was received from a queue, it will be sent to a queue; if the message was received from a topic, it will be sent to a topic.

The argument component **<target>** can be any string and can include the character **$**, which is automatically replaced with the original destination name.

Example 3 shows a string that specifies a 1 second delay following 5 failed attempts at redelivery, a 5 second delay following a total of 10 failed attempts, then redirects the message to a dead-letter queue named **mydlq** after a total of 50 failed attempts.

Example 3

```
5:1000; 10:5000; 50:move(queue:mydlq)
```

Example 4 shows a string that specifies a 1 second delay following 5 failed attempts at redelivery, a 5 second delay following a total of 10 failed attempts, then redirects the message to a dead-letter queue after a total of 50 failed attempts. If the message was received from a source destination named **Queue1**, the message will be redirected to a target destination named **dlqQueue1error**.

Example 4

```
5:1000; 10:5000; 50:move(queue:dlq$error)
```

Delay and Delete

After a certain number of failed redelivery attempts, you may want to simply delete the message. The format for deleting is **<retries:delete>**. Example 5 shows a string that specifies a 1 second delay following 5 failed attempts at redelivery, a 5 second delay following a total of 10 failed attempts, then deletes the message.

Example 5

```
5:1000; 10:5000; 50:delete
```

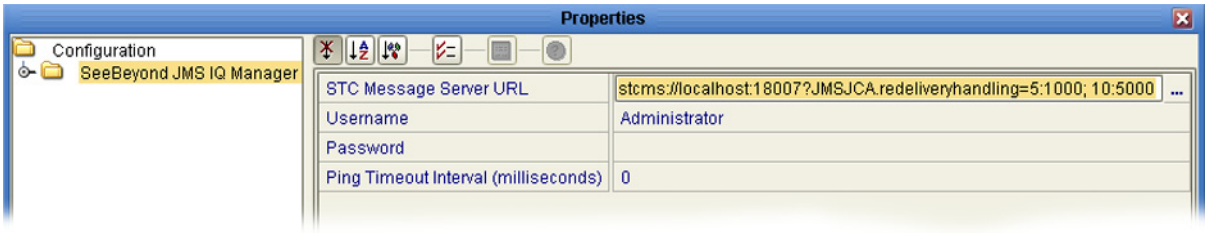
3.5.2 Specifying Redelivery Options in the JMS IQ Manager

You can specify the actions you want to be taken after a message has been rolled back by appending a redelivery-handling string to the message server URL when you configure the JMS IQ Manager. These actions then override the default actions for all JMS clients interacting with the JMS IQ Manager. The format for this string is:

```
?JMSJCA.redeliveryhandling=<action>
```

where <action> is the string specifying the delay/redirection/deletion as given in the examples of the preceding section. Figure 9 illustrates how Example 2 would be specified in the property dialog (see also the *Sun SeeBeyond eGate™ Integrator User's Guide*).

Figure 9 JMS IQ Manager Configuration Properties - Redelivery Example

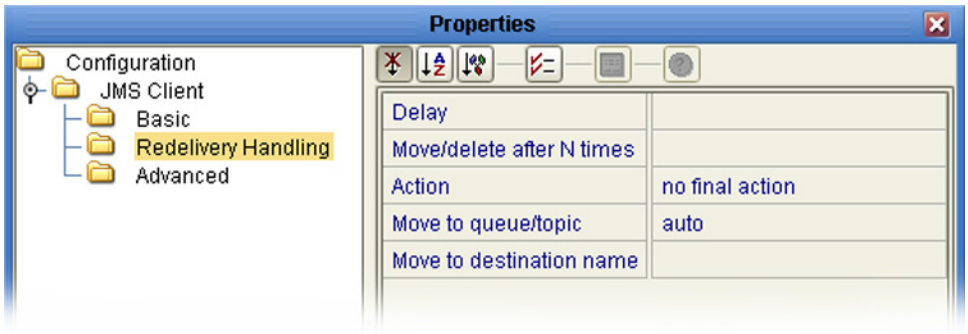


3.5.3 Specifying Redelivery Options in a JMS Client

You can also specify the actions you want taken for a specific JMS client when specifying the configuration properties for that client (see Figure 10). If you do so, the properties you specify for the JMS client will override the redelivery properties specified for the JMS IQ Manager, for that client only.

For descriptions of the JMS client redelivery configuration properties, see **JMS Client Configuration** on page 85, or "Configuring JMS Clients" in the *Projects* chapter of the *Sun SeeBeyond eGate™ Integrator User's Guide*.

Figure 10 JMS Client Configuration Properties - Redelivery Handling



3.6 Enqueued Message Properties

Messages contained within a message destination (queue or topic) are assigned certain properties for the length of time that they reside in the message destination. These properties are displayed when you check the status of the message destination using either Enterprise Manager or the command-line MS Control utility (see **JMS IQ Manager Management** on page 110).

Note that except for when there are no unconsumed messages in the topic or queue, the message having the *first enqueue time* will have the *minimum sequence number* and the message having the *last enqueue time* will have the *maximum sequence number*.

3.6.1 Enqueue Time

Each message is given a message enqueue time by the JMS IQ Manager when a message is added into a queue or topic. It is determined by the server side and is unique across a topic or a queue. For a transaction session, the message enqueue time is determined by the JMS IQ Manager when a message is committed. For a non-transaction session, the message enqueue time is determined by the JMS IQ Manager when the JMS IQ Manager receives the message.

The *first enqueue time* of a topic or queue is the enqueue time of the first unconsumed message in the topic or queue. The *last enqueue time* of a topic or queue is the enqueue time of the last unconsumed message in the topic or queue. If there is no unconsumed message in the topic or queue, then the last enqueue time and the first enqueue time are identical and represent the enqueue time of the most recently consumed message in the topic or queue.

The following example displays the current status of the queue **PTP**:

```
stcmsctrlutil -host localhost -port 24055 -queuestat PTP
Queue Name: PTP
First enqueue time: 03212005:08:33:09
Last enqueue time: 03212005:08:33:10
Number of current receivers: 2
Message count: 4
Messages sent and committed: 245
Min sequence number: 245
Max sequence number: 248
Suspended: No
```

In the above example, there are four messages in the queue **PTP**. The *first enqueue time* is the enqueue time of the first message which is **03212005:08:33:09**, and the *last enqueue time* is the enqueue time of the fourth message which is **03212005:08:33:10**. If all four messages are consumed, both the first and last enqueue times become the enqueue time of the fourth message, which is **03212005:08:33:10**.

In the initial state, where the JMS IQ Manager has not received or consumed any messages in the topic or queue, both the first and last enqueue times are **N/A**.

Note: *You should not change the clock of the computer on which the JMS IQ Manager is running once the JMS IQ Manager is configured. If you do, the message enqueue time will be incorrect.*

3.6.2 Sequence Number

Each message has a message sequence number, which is assigned by the JMS IQ Manager when the message is added to the queue or topic. The message sequence number provides a message index, represented as a long integer value, and is unique across a topic or queue. For a transaction session, the message sequence number is determined by the JMS IQ Manager when a message is committed. For a non-transaction session, the message sequence number is determined by the JMS IQ Manager when the JMS IQ Manager receives the message.

The *minimum sequence number* of a topic or queue is the sequence number of the first unconsumed message in the topic or queue. The *maximum sequence number* of a topic or a queue is the sequence number of the last unconsumed message in the topic or queue. Since all messages in the topic or the queue are sorted and indexed by sequence number, the first unconsumed message has the minimum sequence number, and the last unconsumed message has the maximum sequence number, among those residing in the topic or queue. If there is no unconsumed message in the topic or queue, the minimum sequence number and the maximum sequence number are identical and represent the next available sequence number — which will be assigned to the next message received by the topic or queue.

The following example displays the current status of the queue **PTP**:

```
stcmsctrlutil -host localhost -port 24055 -queuestat PTP
Queue Name: PTP
First enqueue time: 03212005:08:33:09
Last enqueue time: 03212005:08:33:10
Number of current receivers: 2
Message count: 4
Messages sent and committed: 245
Min sequence number: 245
Max sequence number: 248
Suspended: No
```

In the above example, there are four messages in the queue **PTP**. The *minimum sequence number* is the sequence number of the first message which is **245**, and the *maximum sequence number* is the sequence number of the fourth message which is **248**. If all four currently resident messages are consumed, both the minimum and maximum sequence numbers will represent the next available sequence number, which is **249**. This is the number that will be assigned to the next incoming message.

In the initial state, where the JMS IQ Manager has not received or consumed any messages in the topic or queue, both the minimum and maximum sequence numbers are **0** (zero).

3.7 Performance Issues

Because of the large assortment of configuration parameters, you have a high degree of control over processing speed, memory use, and disk space. The JMS IQ Manager properties work together to allow you to fine-tune your system according to load and hardware constraints. For information, refer to **Segment Properties** on page 96.

Because every message is written to disk, file input/output (I/O) is usually the hardware factor with the largest performance impact. For a disk with adequate I/O speed, highest performance is achieved by holding all messages in server memory continuously until the corresponding segment is cleaned up.

Because available server memory can easily be exceeded for systems handling very large messages, there are several configuration parameters to help you manage a memory-bound server; see **Throttling Producers** on page 34.

To maximize performance

- Use the fastest disk possible.
- Allocating a new segment requires more time than freeing a cleaned-up segment.
- Set the segment size to lower values, because smaller segments turn over more rapidly and thus provide more effective use of server memory. However, because cleaning up two small segments requires more time than cleaning up one large segment, you can use very large segments to increase performance on systems that are constrained by disk I/O speed rather than memory or space.

3.7.1 Throttling Producers

In addition to disk and memory-management features provided by the operating system, there are special configuration properties within by the JMS IQ Manager that specifically deal with messages, message destinations, and producers.

When the amount of JMS IQ Manager memory allocated to messages reaches a certain limit, the JMS IQ Manager can be instructed to stop reading all messages from one or more producers until certain criteria are met. This process is referred to as *throttling* the producer.

Producer throttling is done on a per-message destination basis. This addresses the two most common reasons for approaching the JMS IQ Manager memory limit in an otherwise well-tuned system:

- **A particular message destination has a period of abnormally heavy traffic.**

  Throttling all producers that feed the message destination gives the destination's consumer a chance to catch up while maintaining normal throughput for other message destinations.

- **A particular consumer fails, causing a backup of all message destinations to which it subscribes.**

  If the consumer problem is transient in nature, then throttling all its producers gives it a chance to catch up on the backlog. If the consumer problem is permanent, then

throttling its producers allows unaffected message destinations to flow freely while the problem can be diagnosed and repaired without taking the server offline.

Three configuration properties govern producer throttling:

- The **Server Throttling Threshold** property sets the *message server* limit. When the JMS IQ Manager is below this threshold, it does not throttle any producers — regardless of whether they are feeding a destination that has exceeded the Per-Destination Throttling Threshold.
- When the Server Throttling Threshold has been exceeded, and producer throttling is in effect, the **Per-Destination Throttling Threshold** property specifies the *per-destination* limit. The JMS IQ Manager stops reading messages from producers feeding any message destination that has exceeded this limit.
- **Throttling lag** determines how many messages for this topic must be removed from the queue before throttling can stop.

Once throttling has been initiated, the JMS IQ Manager resumes reading messages for the affected message destinations only when one of the following criteria is met:

- The JMS IQ Manager falls below the **Server Throttling Threshold** limit.
- The number of undelivered messages in the affected destination has dropped to less than the difference between the value specified by the **Per-Destination Throttling Threshold** and the value specified by the **Throttling Lag**.

Note: *Each message in a message destination counts against the message destination's **Per-Destination Throttling Threshold** limit until the message is dequeued. In particular: A non-transactional message is counted until it has been delivered to all its subscribers; a transactional or XA-compliant message is counted until it has been committed by all consumers.*

Example of Producer Throttling and Unthrottling

Table 8 illustrates a scenario where a JMS IQ Manager exceeds its specified threshold and begins to throttle selected producers. In this example, the JMS IQ Manager uses the following (default) values for throttling properties:

- **Server Throttling Threshold** = 100,000
- **Per-Destination Throttling Threshold** = 1,000
- **Throttling lag** = 100

Two minutes later, this affects Topic_A, which has two subscribers and one publisher: Its publisher is throttled for three minutes, until the number of undelivered messages can drop below 900. Later, because the JMS IQ Manager is no longer loaded, the same topic is allowed build up an even greater backlog without having its publisher throttled.

Table 8 Publisher Throttling Example

Time	For Server: Total Messages on All Topics	For Messages in Topic_A (only): The Highest Sequence Number			Comment
		Read from Pub1:	Sent to Sub1:	Sent to Sub2:	
11:37	98604	500	200	75	Server is not yet over limit.
11:38	100307	800	500	150	Server is now over limit, but Topic_A is unaffected — its subscribers are keeping up well enough.
11:39	101283	1100	800	225	Server still over limit, Topic_A still unaffected — only 875 undelivered messages.
11:40	103429	1350	1050	300	Topic_A has crossed the per-destination limit now that it has 1050 undelivered messages; while the server remains over limit, Pub1 will stay throttled until the number of undelivered messages falls below 900 (the difference between the value specified by the *Per-Destination Throttling Threshold* and the value specified by the *Throttling Lag*).
11:41	104031	1350	1300	375	Pub1 is throttled; Sub1 is nearly caught up; Sub2 is catching up, but has 975 undelivered messages.
11:42	103204	1350	1350	449	Pub1 is throttled; Sub1 has caught up; Sub2 has 901 undelivered messages — still too many.
11:43	102762	1350	1350	451	Although server is still over limit, it unthrottles Pub1 now that the undelivered message count for Topic_A has fallen below 900.
11:44	101095	1375	1370	525	Server is over limit, but Topic_A is unaffected — it has only 850 undelivered messages.
11:45	100028	1575	1500	600	Server is over limit, but Topic_A is unaffected — it has only 975 undelivered messages.
11:46	99248	1900	1700	675	Server is no longer over limit; no publishers are throttled even though Sub2 has more than 1000 undelivered messages.

Chapter 4

The JMS OTD

The JMS OTD describes features of the JMS OTD, and the procedure for setting JMS properties in Java-based Collaboration Definitions.

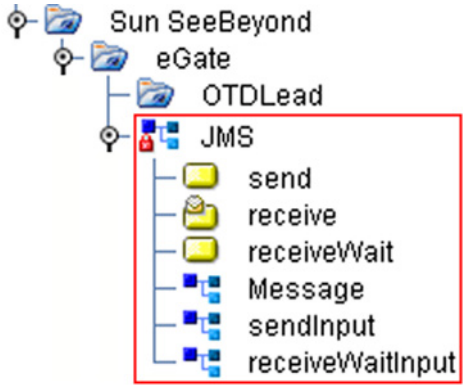
What's in This Chapter

4.1 About the JMS OTD

The Java Message Service (JMS) OTD is a special type of OTD that allows Collaborations to read from and write to topics or queues. It supplies operators and Java methods for creating, sending, and receiving JMS messages (see **JMS OTD Methods** on page 43).

The JMS OTD is included with Sun SeeBeyond eGate Integrator, and is installed automatically. The template resides in the **Sun Seebeyond\eGate** folder in Project Explorer (see Figure 11).

Figure 11 Project Explorer - JMS OTD



The **send**, **receive**, and **receiveWait** nodes represent *web service operations* that are available in the Collaboration Definition Wizard (Java) when you create a Java-based Collaboration Definition implementing an existing web service. The **Message**, **sendInput**, and **receiveWaitInput** nodes represent *messages* used with these operations, as listed in Table 9.

Table 9 JMS Web Service Messages

Message Node	Function
Message	Message for **receive** operation.
sendInput	Message for **send** operation.
receiveWaitInput	Message for **receiveWait** operation.

When you select one of the web service operations for a Java-based Collaboration Definition implementing an *existing* web service, the corresponding message becomes available in the Collaboration Definition Editor as the *input* argument. When you create a Java-based Collaboration Definition implementing a *new* web service, the three messages become available as both *input* and *output* arguments.

The **send**, **receive**, and **receiveWait** nodes can also be dragged and dropped into the eInsight Business Process Editor as *activities*. The corresponding messages are then displayed in the mapper while performing assigns. See the *Sun SeeBeyond eInsight™ Business Process Manager User's Guide* for additional information.

4.1.1 Message Types

The JMS OTD currently supports the following message types:

- **Message**

  A *Message* carries no payload, and is generally used for event notification. Java methods specifically available for use with this message type are:

  - **createMessage()** on page 46
  - **createMessage(msg)** on page 47

- **BytesMessage**

  A *BytesMessage* carries a byte array as its payload, and is often used in cases where JMS is simply used as a transport between systems. Java methods specifically available for use with this message type are:

  - **createBytesMessage()** on page 45
  - **createBytesMessage(msg)** on page 45
  - **getBytesMessage()** on page 68
  - **retrieveBytesFromMessage()** on page 74
  - **retrieveBytesFromMessage(arg0)** on page 75
  - **setBytesMessage(arg0)** on page 79

- **MapMessage**

  A *MapMessage* carries a set of name-value pairs as its payload, and is often used for delivering keyed data. Java methods specifically available for use with this message type are:

  - **createMapMessage()** on page 46

- **countMapMessage()** on page 67
- **getMapMessage()** on page 70
- **getMapMessage(arg0)** on page 70
- **retrieveMapMessage(arg0)** on page 76
- **retrieveMapMessageList()** on page 76
- **storeMapMessage(arg0, arg1)** on page 82

- **StreamMessage**

  A *StreamMessage* carries a stream of primitive Java types (such as **char**, **double**, and **int**) as its payload, and is often used when delivering primitive application data in a fixed order. Java methods specifically available for use with this message type are:

  - **createStreamMessage()** on page 47
  - **countStreamMessage()** on page 67
  - **getStreamMessage()** on page 71
  - **setStreamMessage(arg0, arg1)** on page 81

- **TextMessage**

  A *TextMessage* carries a string (of type **java.lang.String**) as its payload, and is used for exchanging both text messages and XML documents. As such, it is the most often used message type. Java methods specifically available for use with this message type are:

  - **createTextMessage()** on page 48
  - **createTextMessage(msg)** on page 48
  - **getTextMessage()** on page 72
  - **retrieveStringFromMessage(arg0)** on page 77
  - **retrieveStringFromMessage()** on page 78
  - **setTextMessage(arg0)** on page 81

4.2 JMS Message Properties

You can set specific message properties in the JMS OTD using the property nodes that are exposed by expanding the appropriate nodes in the Collaboration Definition Editor.

4.2.1 JMS Message Header Properties

For inbound or outbound JMS messages, you can set the message header properties shown in Figure 12 (the same properties are available for the *output* node in the **receiveWait** operation). When you set these properties in a Java-based Collaboration Definition, they are used only by the Collaboration that uses that specific Collaboration Definition.

Figure 12 JMS Message Property Nodes (receive operation)

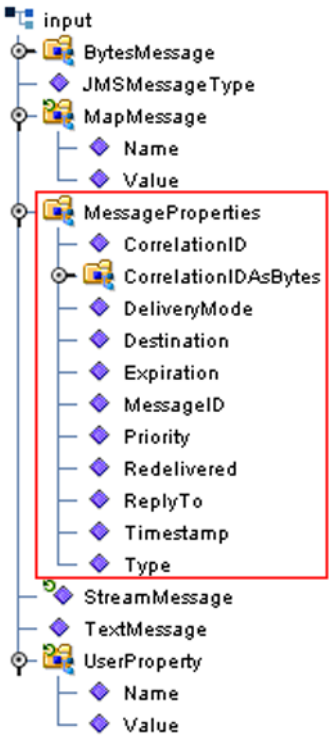


Table 10 shows the allowed values for the properties and the JMS methods used when you set a property in a Collaboration. The methods that are exposed for your use are cross-referenced to the appropriate description. The methods that are not cross-referenced are automatically assigned, and not exposed.

Table 10 JMS Message Header Properties

Property	Values	Equivalent JMS Methods
CorrelationID	correlation ID	getJMSCorrelationID() setJMSCorrelationID(string)

Property	Values	Equivalent JMS Methods
CorrelationIDAsBytes	correlation ID	getCorrelationIDAsBytes setCorrelationIDAsBytes(byte[])
DeliveryMode	persistent, nonpersistent (default = persistent)	**getDeliveryMode()** on page 49 **setDeliveryMode(arg0)** on page 62
Destination	destination (default = message destination as configured in Connectivity Map)	**getDestination()** on page 49 **setDestination(arg0)** on page 63
Expiration	number in milliseconds (default = 86,400,000 ms = 24 hours)	**getTimeToLive()** on page 51 **setTimeToLive(arg0)** on page 64
MessageID	message ID	getJMSMessageID() setJMSMessageID(string)
Priority	0 - 9 where 9 is the highest priority (default = 4)	**getPriority()** on page 50 **setPriority(arg0)** on page 64
Redelivered	true, false	getJMSRedelivered() setJMSRedelivered(boolean)
ReplyTo	destination	getJMSReplyTo() setJMSReplyTo(destination)
Timestamp	number in milliseconds	getJMSTimestamp() setJMSTimestamp(long)
Type	Text, Bytes, Map, Stream	**getJMSMessageType()** on page 69 **setJMSMessageType(arg0)** on page 80

4.2.2 Additional JMS Message Properties

For outbound JMS messages, you can also set the message properties shown in Figure 13, which determine only how the message is sent. When you set these properties in a Java-based Collaboration Definition, they override the corresponding JMS client message priorities, and are used only by the Collaboration that uses that specific Collaboration Definition. For information on setting these properties in the JMS clients, see "Configuring JMS Clients" in the *Projects* chapter of the *Sun SeeBeyond eGate™ Integrator User's Guide*.

Figure 13 JMS OTD Outbound Property Nodes

Table 11 shows the allowed values for the outbound JMS message properties and the JMS methods used when you set a property in a Collaboration.

Table 11 JMS OTD Outbound Properties

Property	Allowed Values	Equivalent JMS Methods
deliveryMode	persistent, nonpersistent	**getDeliveryMode()** on page 49 **setDeliveryMode(arg0)** on page 62
priority	0 - 9 where 9 is the highest priority	**getPriority()** on page 50 **setPriority(arg0)** on page 64
timeToLive	number in milliseconds	**getTimeToLive()** on page 51 **setTimeToLive(arg0)** on page 64
destination	destination	**getDestination()** on page 49 **setDestination(arg0)** on page 63
MessageServerURL	A valid URL	**getMessageServerURL()** on page 50 **setMessageServerURL(arg0)** on page 63

For more information about these properties, refer to the equivalent JMS method description in the indicated locations.

Chapter 5

JMS OTD Methods

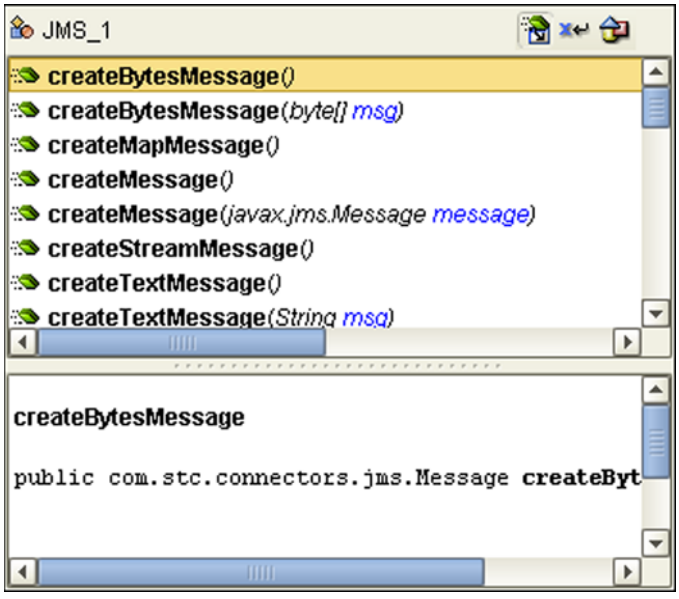
JMS OTD Methods describes Java methods available for use with the JMS OTD.

What's in This Chapter

5.1 Using the JMS OTD in Collaboration Definitions

To allow Java-based Collaborations to read from and write to topics or queues, you must add the JMS OTD to the Collaboration Definition, as described in the *Sun SeeBeyond eGate™ Integrator User's Guide*. The JMS OTD contains a set of Java methods that are used by the JMS client to manage the JMS session. These methods are accessed by right-clicking the OTD root node in the Collaboration Definition Editor to display the context menu, and then clicking **Select method to call …** to display the method browser (see Figure 14). You can also drag the node to the mapping area to display the browser.

Figure 14 Java Method Browser



5.2 JMS Java Methods

This section describes the Java methods available for use with objects of the type **com.stc.connectors.jms.JMS**, and can be accessed from:

- The JMS OTD.
- Input messages for the **send** web service operation.
- Output messages for the **receive** web service operation (new web service).

Table 12 JMS Methods

createBytesMessage()

Description

Creates an empty byte message.

Parameters

None.

Return Value

Returns	Type
The byte message object.	com.stc.connectors.jms.Message

Exceptions

None.

createBytesMessage(msg)

Description

Creates a byte message with the specified byte array value.

Parameters

Name	Type	Description
msg	byte[]	byte array value for the bytes message

Return Value

Returns	Type
The message object with data *msg*.	com.stc.connectors.jms.Message

Exceptions

None.

createMapMessage()

Description

Creates a map message.

Parameters

None.

Return Value

Returns	Type
The map message object.	com.stc.connectors.jms.Message

Exceptions

None.

createMessage()

Description

Creates an empty message (no payload). This is the most efficient method for event notification.

Parameters

None.

Return Value

Returns	Type
The message object.	com.stc.connectors.jms.Message

Exceptions

None.

createMessage(msg)

Description

Creates a message of type **com.stc.connectors.jms.Message**, which wraps around the message variable of type **javax.jms.Message**.

Parameters

Name	Type	Description
msg	javax.jms.Message	The message variable.

Return Value

Returns	Type
The message object, with data from JMS message object.	com.stc.connectors.jms.Message

Exceptions

Throws **JMSException**, **IOException**.

createStreamMessage()

Description

Creates an empty stream message. Use the method **setStreamMessage(arg0, arg1)** on page 81 to add the payload.

Parameters

None.

Return Value

Returns	Type
The stream message object.	com.stc.connectors.jms.Message

Exceptions

None.

createTextMessage()

Description

Creates an empty text message. Use the method **setTextMessage(arg0)** on page 81 to add the payload.

Parameters

None.

Return Value

Returns	Type
The text message object.	com.stc.connectors.jms.Message

Exceptions

None.

createTextMessage(msg)

Description

Creates a text message that includes the specified text, producing a ready-to-deliver **TextMessage** object.

Parameters

Name	Type	Description
msg	String	The string to populate the *msg* object with.

Return Value

Returns	Type
The message object with data *msg*.	com.stc.connectors.jms.Message

Exceptions

None.

getDeliveryMode()

Description

Gets and returns the value of the JMS delivery mode.

Parameters

None.

Return Value

Returns	Type
The value of the JMS *deliveryMode* property.	java.lang.String

Exceptions

None.

getDestination()

Description

Gets and returns the name of the queue or topic.

Parameters

None.

Return Value

Returns	Type
The message destination name.	java.lang.String

Exceptions

None.

getMessageServerURL()

Description

Gets and returns the value of the message server URL.

Parameters

None.

Return Value

Returns	Type
The value of the message server URL.	java.lang.String

Exceptions

None.

getPriority()

Description

Gets and returns the value of the JMS priority.

Parameters

None.

Return Value

Returns	Type
The value of the JMS *priority* property.	int

Exceptions

None.

getTimeToLive()

Description

Gets and returns the specified time period, in milliseconds, following the dispatch time that a message should be retained by the message system.

Parameters

None.

Return Value

Returns	Type
The default time period of the JMS *timeToLive* property, in milliseconds.	long

Exceptions

None.

receive(timeout)

Description

Receives the next message of type **com.stc.connectors.jms.Message** that arrives from the destination configured in the Connectivity Map Editor, during the specified timeout interval.

Parameters

Name	Type	Description
timeout	long	The number of milliseconds before the *receive* method times out.

Return Value

Returns	Type
The next message produced, or null if the timeout expires.	com.stc.connectors.jms.Message

Exceptions

Throws **JMSException**.

receive(timeout, destination)

Description

Receives the next message of type **com.stc.connectors.jms.Message** that arrives from the specified message destination during the specified timeout interval.

Parameters

Name	Type	Description
timeout	long	The number of milliseconds before the *receive* method times out.
destination	java.lang.String	The name of the topic or queue from which this method receives messages.

Return Value

Returns	Type
The next message produced, or null if the timeout expires.	com.stc.connectors.jms.Message

Exceptions

Throws **JMSException**.

receiveNoWait()

Description

Receives the next message of type **com.stc.connectors.jms.Message**, if one is immediately available.

Parameters

None.

Return Value

Returns	Type
The next message produced, or null if there is no message available.	com.stc.connectors.jms.Message

Exceptions

Throws **JMSException**.

receiveNoWait(destination)

Description

Receives the next message of type **com.stc.connectors.jms.Message** from the specified message destination, if a message is immediately available.

Parameters

Name	Type	Description
destination	java.lang.String	The name of the topic or queue from which this method receives messages.

Return Value

Returns	Type
The next message produced, or null if there is no message available.	com.stc.connectors.jms.Message

Exceptions

Throws **JMSException**.

requestReply(message)

Description

Sends a message of type **com.stc.connectors.jms.Message** to the destination configured in the Connectivity Map Editor, and waits for the reply message by using **Topic/QueueRequestor**.

Parameters

Name	Type	Description
message	com.stc.connectors .jms.Message	The message object to send

Return Value

Returns	Type
The reply message.	com.stc.connectors.jms.Message

Exceptions

Throws **JMSException** when a message is null or when the JMS provider fails to send and receive the message due to an internal error.

requestReply(timeout, message)

Description

Sends a message of type **com.stc.connectors.jms.Message** to the destination configured in the Connectivity Map Editor, and receives the reply message by using **Topic/QueueRequestor** during the specified timeout interval.

Parameters

Name	Type	Description
timeout	java.lang.long	The timeout in milliseconds
message	com.stc.connectors.jms.Message	The message object to send

Return Value

Returns	Type
The reply message, if available during the timeout interval.	com.stc.connectors.jms.Message

Exceptions

Throws **JMSException** if a message is null, JMS provider fails to send and receive the message due to an internal error, or timeout expires.

requestReplyTo(message, destName)

Description

Sends a message of type **com.stc.connectors.jms.Message** to the explicitly named destination, and waits for the reply message by using **Topic/QueueRequestor**.

Parameters

Name	Type	Description
message	com.stc.connectors.jms.Message	The message object to send
destName	java.lang.String	The destination name

Return Value

Returns	Type
The reply message.	com.stc.connectors.jms.Message

Exceptions

Throws **JMSException** when a message is null, **destName** is null, or the JMS provider fails to send and receive the message due to an internal error.

requestReplyTo(timeout, message, destName)

Description

Sends a message of type **com.stc.connectors.jms.Message** to the explicitly named destination, and receives the reply message by using **Topic/QueueRequestor**. Specified timeout is to be applied.

Parameters

Name	Type	Description
timeout	java.lang.long	The timeout in milliseconds
message	com.stc.connectors .jms.Message	The message object to send
destName	java.lang.String	The destination name

Return Value

Returns	Type
The reply message, if available during the timeout interval.	com.stc.connectors.jms.Message

Exceptions

Throws **JMSException** when a message is null, **destName** is null, or when the JMS provider fails to send and receive the message due to an internal error, or timeout expires.

send(message)

Description

Sends a message of type **com.stc.connectors.jms.Message** to the destination configured in the Connectivity Map Editor, using the JMS session's default settings for message priority, time to live, and delivery mode.

Parameters

Name	Type	Description
message	com.stc.connectors .jms.Message	The message variable type to be sent.

Return Value

None.

Exceptions

Throws **JMSException** if a message is null or if the JMS provider fails to send the message due to an internal error.

send(message, deliveryMode, priority, timeToLive)

Description

Sends a message of type **com.stc.connectors.jms.Message** to the destination configured in the Connectivity Map Editor, using the specified parameters.

Parameters

Name	Type	Description
message	com.stc.connectors .jms.Message	The message variable to be created.
deliveryMode	int	The message delivery mode; 1 indicates nonpersistent messages, 2 indicates persistent messages.
priority	int	The message priority (0 through 9, with 9 being the highest priority).
timeToLive	long	The amount in milliseconds before the message expires.

Return Value

None.

Exceptions

Throws **JMSException** if a message is null or if the JMS provider fails to send the message due to an internal error.

sendBytes(payload)

Description

Sends a message of type **byte[]** to the destination configured in the Connectivity Map Editor, using the JMS session's default settings for message priority, time to live, and delivery mode.

Parameters

Name	Type	Description
payload	*byte[]*	The message byte array value.

Return Value

None.

Exceptions

Throws **JMSException** if a message is null or if the JMS provider fails to send the message due to an internal error.

sendBytes(payload, deliveryMode, priority, timeToLive)

Description

Sends a message of type **byte[]** to the destination configured in the Connectivity Map Editor, using the specified parameters.

Parameters

Name	Type	Description
payload	*byte[]*	The byte array value.
deliveryMode	int	The message delivery mode; 1 indicates nonpersistent messages, 2 indicates persistent messages.
priority	int	The message priority (0 through 9, with 9 being the highest priority).
timeToLive	long	The amount in milliseconds before the message expires.

Return Value

None.

Exceptions

Throws **JMSException** if a message is null or if the JMS provider fails to send the message due to an internal error.

sendBytesTo(payload, destination)

Description

Sends a message of type **byte[]** to the specified destination, using the JMS session's default settings for message priority, time to live, and delivery mode.

Parameters

Name	Type	Description
payload	*byte[]*	The byte array value.
destination	java.lang.String	The name of the topic or queue from which this method receives messages.

Return Value

None.

Exceptions

Throws **JMSException** if a message is null or if the JMS provider fails to send the message due to an internal error.

sendBytesTo(payload, destination, deliveryMode, priority, timeToLive)

Description

Sends a message of type **byte[]** to the destination configured in the Connectivity Map Editor, using the specified parameters.

Parameters

Name	Type	Description
payload	byte[]	The byte array value.
deliveryMode	int	The message delivery mode; 1 indicates nonpersistent messages, 2 indicates persistent messages.
priority	int	The message priority (0 through 9, with 9 being the highest priority).
timetolive	long	The amount in milliseconds before the message expires.

Return Value

None.

Exceptions

Throws **JMSException** if a message is null or if the JMS provider fails to send the message due to an internal error.

sendText(payload)

Description

Sends a message of type **java.lang.String** to the destination configured in the Connectivity Map Editor, using the JMS session's default settings for message priority, time to live, and delivery mode.

Parameters

Name	Type	Description
payload	java.lang.String	The text in the message.

Return Value

None.

Exceptions

Throws **JMSException** if a message is null or if the JMS provider fails to send the message due to an internal error.

sendText(payload, deliveryMode, priority, timeToLive)

Description

Sends a message of type **java.lang.String** to the destination configured in the Connectivity Map Editor, using the specified parameters.

Parameters

Name	Type	Description
payload	java.lang.String	The text in the message.
deliveryMode	int	The message delivery mode; 1 indicates nonpersistent messages, 2 indicates persistent messages.
priority	int	The message priority (0 through 9, with 9 being the highest priority).
timeToLive	long	The amount in milliseconds before the message expires.

Return Value

None.

Exceptions

Throws **JMSException** if a message is null or if the JMS provider fails to send the message due to an internal error.

sendTextTo(payload, destination)

Description

Sends a message of type **java.;ang.String** to the specified destination, using the JMS session's default settings for message priority, time to live, and delivery mode.

Parameters

Name	Type	Description
payload	java.lang.String	The text in the message.
destination	java.lang.String	The name of the topic or queue to which this method should send the message.

Return Value

None.

Exceptions

Throws **JMSException** if a message is null or if the JMS provider fails to send the message due to an internal error.

sendTextTo(payload, destination, deliveryMode, priority, timeToLive)

Description

Sends a message of type **java.lang.String** to the specified destination, using the specified parameters.

Parameters

Name	Type	Description
payload	java.lang.String	The text in the message.
destination	java.lang.String	The name of the topic or queue to which this method should send the message.
deliveryMode	int	The message delivery mode; 1 indicates nonpersistent messages, 2 indicates persistent messages.
priority	int	The message priority (0 through 9, with 9 being the highest priority).
timeToLive	long	The amount in milliseconds before the message expires.

Return Value

None.

Exceptions

Throws **JMSException** if a message is null or if the JMS provider fails to send the message due to an internal error.

sendTo(message, destination)

Description

Sends a message of type **com.stc.connectors.jms.Message** to the specified destination, using the JMS session's default settings for message priority, time to live, and delivery mode.

Parameters

Name	Type	Description
msg	com.stc.connectors .jms.Message	The message to be sent.
destination	java.lang.String	The name of the topic or queue to which this method should send the message.

Return Value

None.

Exceptions

Throws **JMSException** if a message is null or if the JMS provider fails to send the message due to an internal error.

sendTo(message, destination, deliveryMode, priority, timeToLive)

Description

Sends a message of type **com.stc.connectors.jms.Message** to the specified destination, using the specified parameters.

Parameters

Name	Type	Description
message	com.stc.connectors .jms.Message	The message variable type.
destination	java.lang.String	The name of the topic or queue from which this method receives messages.
deliveryMode	int	The message delivery mode; 1 indicates nonpersistent messages, 2 indicates persistent messages.
priority	int	The message priority (0 through 9, with 9 being the highest priority).
timeToLive	long	The amount in milliseconds before the message expires.

Return Value

None.

Exceptions

Throws **JMSException** if a message is null or if the JMS provider fails to send the message due to an internal error.

setDeliveryMode(arg0)

Description

Sets the value of the JMS delivery mode.

Parameters

Name	Type	Description
arg0	java.lang.String	The value of the JMS *deliveryMode* property.

Return value

None.

Exceptions

None.

setDestination(arg0)

Description

Specifies the queue or topic.

Parameters

Name	Type	Description
arg0	java.lang.String	The message destination name.

Return value

None.

Exceptions

None.

setMessageServerURL(arg0)

Description

Sets the value of the message server URL.

Parameters

Name	Type	Description
arg0	java.lang.String	The value of the message server URL.

Return value

None.

Exceptions

None.

setPriority(arg0)

Description

Sets the value of the JMS priority.

Parameters

Name	Type	Description
arg0	int	The value of the JMS *priority* property.

Return value

None.

Exceptions

None.

setTimeToLive(arg0)

Description

Sets the default time period, in milliseconds, following the dispatch time that a message should be retained by the message system.

Parameters

Name	Type	Description
arg0	long	The default time period for the JMS *timeToLive* property, in milliseconds.

Return value

None.

Exceptions

None.

Chapter 6

JMS Message Methods

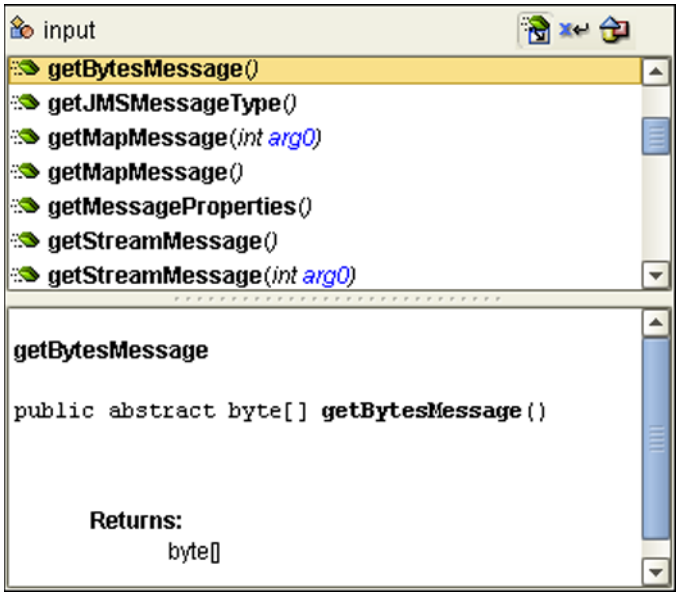
JMS Message Methods describes Java methods available for use with JMS messages in Java-based Collaboration Definitions.

What's in This Chapter

6.1 Using JMS Messages in Collaboration Definitions

To create Java Collaborations that implement an existing web service, you add the JMS **receive**, **receiveWait**, or **send** web service operation to the Collaboration Definition, as described in the *Sun SeeBeyond eGate™ Integrator User's Guide*. Each of these operations has a set of Java methods that is available for use with it. To access these methods, right-click the **input** or **output** node in the Collaboration Definition Editor to display the context menu, and click **Select method to call …** to display the method browser (see Figure 15). You can also drag the node to the mapping area to display the browser.

Figure 15 Java Method Browser



6.2 Java Methods for JMS Messages

This section describes Java methods that are available for messages of the type **com.stc.connectors.jms.Message**, and can be accessed from:

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Included in the list are methods available for input messages for the **receiveWait** web service operation, which are of the type **com.stc.connectors.jms.receiveWaitMessage**.

Table 13 JMS Message Methods

countMapMessage()

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Counts the number of keys in the map message.

Parameters

None.

Return Value

Returns	Type
The number of keys in the map message.	int

Exceptions

None.

countStreamMessage()

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Counts the number of items in the map message.

Parameters

None.

Return Value

Returns	Type
The number of items in the stream message.	int

Exceptions

None.

countUserProperty()

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Counts the number of user properties.

Parameters

None.

Return Value

Returns	Type
The number of user properties.	int

Exceptions

None.

getBytesMessage()

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Gets the data bytes in the message.

Parameters

None.

Return Value

Returns	Type
The data bytes in the message.	byte []

Exceptions

None.

getDestination()

Available For

- Input messages for the **receiveWait** web service operation.

Description

Gets and returns the name of the queue or topic.

Parameters

None.

Return Value

Returns	Type
The message destination name.	java.lang.String

Exceptions

None.

getJMSMessageType()

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Gets the JMS message type.

Parameters

None.

Return Value

Returns	Type
The JMS message type (for example: *Bytes*, *Text*, *Stream*, *Map*).	java.lang.String

Exceptions

None.

getMapMessage()

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Gets the name-value pair array containing the map message.

Parameters

None.

Return Value

Returns	Type
The NameValuePair array containing the map message.	com.stc.connectors.jms.NameValuePair

Exceptions

None.

getMapMessage(arg0)

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Gets the name-value pair containing the map message from the location in the map list specified by *arg0*.

Parameters

Name	Type	Description
arg0	int	The location of the name-value pair in the map message list.

Return Value

Returns	Type
The NameValuePair at the specified location.	com.stc.connectors.jms.NameValuePair

Exceptions

None.

getMessageProperties()

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Gets the message property object to query for the various JMS message properties.

Parameters

None.

Return Value

Returns	Type
The message property object to query for the various JMS message properties.	com.stc.connectors.jms.MessageProperty

Exceptions

None.

getStreamMessage()

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Gets the array of stream message objects.

Parameters

None.

Return Value

Returns	Type
An array of StreamMessage objects.	java.lang.Object

Exceptions

None.

getStreamMessage(arg0)

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Gets the stream message object from the location specified by *arg0*.

Parameters

Name	Type	Description
arg0	int	The location of the stream message in the list.

Return Value

Returns	Type
The StreamMessage object at the specified location.	java.lang.Object

Exceptions

None.

getTextMessage()

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Gets the data string in the message.

Parameters

None.

Return Value

Returns	Type
The data string in the message.	java.lang.String

Exceptions

None.

getTimeToWait()

Available For

- Input messages for the **receiveWait** web service operation.

Description

Gets and returns the timeout period during which the operation blocks program execution, waiting for a message to arrive.

Parameters

None.

Return Value

Returns	Type
The time to wait for a message, in milliseconds.	long

Exceptions

None.

getUserProperty()

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Gets the name-value pair array of the user properties in the message.

Parameters

None.

Return Value

Returns	Type
The NameValuePair array of the user properties in the message.	com.stc.connectors.jms.NameValuePair

Exceptions

None.

getUserProperty(arg0)

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Gets the name-value pair representing user properties from the location specified by *arg0*.

Parameters

Name	Type	Description
arg0	int	Location of the user property in the list.

Return Value

Returns	Type
The NameValuePair from the specified location.	com.stc.connectors.jms.NameValuePair

Exceptions

None.

retrieveBytesFromMessage()

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Returns the byte array corresponding to the payload of the message object.

Parameters

None.

Return Value

Returns	Type
The byte array corresponding to the payload of the message object.	byte[]

Exceptions

Throws **JMSException** if conversion is not possible (for example, cannot convert from a MapMessage to a byte array).

retrieveBytesFromMessage(arg0)

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Returns the byte array corresponding to the message payload, with the encoding as specified by *arg0*.

Parameters

Name	Type	Description
arg0	java.lang.String	The encoding to use when converting to a byte array.

Return Value

Returns	Type
The byte array corresponding to the message payload, with the specified encoding.	byte[]

Exceptions

- Throws **JMSException** if conversion is not possible (for example, cannot convert from a MapMessage to a byte array).
- Throws **UnsupportedEnclodingException** if the name charset is not supported.

retrieveMapMessage(arg0)

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Returns the value of the message specified by *arg0*, or null if the message does not exist.

Parameters

Name	Type	Description
arg0	java.lang.String	The name of the map message.

Return Value

Returns	Type
The value of the specified message, or null if the message does not exist.	java.lang.Object

Exceptions

None.

retrieveMapMessageList()

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Returns a map message list object, which contains map message objects as an array of name-value pairs.

Parameters

None.

Return Value

Returns	Type
A MapMessageList object, which contains map message objects as name-value pairs.	com.stc.connectors.jms.MapMessageList

Exceptions

None.

retrieveStringFromMessage(arg0)

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Returns a string representation of the message payload, with the encoding as specified by *arg0*.

Parameters

Name	Type	Description
arg0	java.lang.String	The encoding to use when converting to a string object.

Return Value

Returns	Type
The string corresponding to the message payload, with the specified encoding.	java.lang.String

Exceptions

- Throws **JMSException** if conversion is not possible (for example, cannot convert from a MapMessage to a String).
- Throws **UnsupportedEnclodingException** if the name charset is not supported.

retrieveStringFromMessage()

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Returns a string representation of the message payload.

Parameters

None.

Return Value

Returns	Type
A string corresponding to the message payload.	java.lang.String

Exceptions

Throws **JMSException** if conversion is not possible (for example, cannot convert from a MapMessage to a String).

retrieveUserProperty(arg0)

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Returns the value of the user-defined string property specified by *arg0*, or null if the property does not exist.

Parameters

Name	Type	Description
arg0	java.lang.String	The name of the user property.

Return Value

Returns	Type
The value of the specified user property, or null if the property does not exist.	java.lang.String

Exceptions

None.

retrieveUserPropertyList()

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Returns the user property list.

Parameters

None.

Return Value

Returns	Type
A UserPropertyList object, which contains user properties as name-value pairs.	com.stc.connectors.jms.UserPropertyList

Exceptions

None.

setBytesMessage(arg0)

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Sets the bytes message to the value specified by *arg0*.

Parameters

Name	Type	Description
arg0	byte[]	The byte array containing the message.

Return Value

None.

Exceptions

None.

setDestination(arg0)

Available For

- Input messages for the **receiveWait** web service operation.

Description

Specifies the queue or topic.

Parameters

Name	Type	Description
arg0	java.lang.String	The message destination name.

Return value

None.

Exceptions

None.

setJMSMessageType(arg0)

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Sets the text message type to the value specified by *arg0*.

Parameters

Name	Type	Description
arg0	java.lang.String	The value to which to set the JMS message type (for example: *Bytes*, *Text*, *Stream*, *Map*).

Return Value

None.

Exceptions

None.

setStreamMessage(arg0, arg1)

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Sets the stream message specified by *arg0* to the value specified by *arg1*.

Parameters

Name	Type	Description
arg0	int	The index of the stream message to be set.
arg1	java.lang.Object	The value to which to set the stream message.

Return Value

None.

Exceptions

None.

setTextMessage(arg0)

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Sets the text message to the value specified by *arg0*.

Parameters

Name	Type	Description
arg0	java.lang.String	The value to which to set the text message.

Return Value

None.

Exceptions

None.

setTimeToWait(arg0)

Available For

- Input messages for the **receiveWait** web service operation.

Description

Sets the timeout period during which the operation blocks program execution, waiting for a message to arrive.

Parameters

Name	Type	Description
arg0	long	The time, in milliseconds, for the operation to block program execution.

Return Value

None.

Exceptions

None.

storeMapMessage(arg0, arg1)

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Writes the name and value of a map message to the map message object, where *arg0* specifies the name and *arg1* specifies the value of the map message.

Parameters

Name	Type	Description
arg0	java.lang.String	The name of the map message.
arg1	java.lang.Object	The value of the map message.

Return Value

None.

Exceptions

None.

storeUserProperty(arg0, arg1)

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

Writes the name and value of a user property to the user property object, where *arg0* specifies the name and *arg1* specifies the value of the user property.

Parameters

Name	Type	Description
arg0	java.lang.String	The name of the user property.
arg1	java.lang.String	The value of the user property.

Return Value

None.

Exceptions

None.

deserialize(arg0)

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

This is an internally-used method.

duplicate()

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

This is an internally-used method.

serialize(arg0)

Available For

- Input messages for the **receive** web service operation.
- Output messages for the **receiveWait** web service operation.

Description

This is an internally-used method.

Chapter 7

JMS Client Configuration

JMS Client Configuration describes the configuration properties for JMS Client connectors.

What's in This Chapter

7.1 Configuration Property Categories

The following tables list the JMS client configuration properties as displayed in the Enterprise Designer configuration dialogs. They serve as a cross-reference to the property descriptions in the following section, which are categorized by *consumer* and *producer*, and presented alphabetically.

7.1.1 Root Properties

Property	Applies to	Description
Durable Subscriber Name	Topic subscribers	**Durable Subscriber Name** on page 88

7.1.2 Basic Properties

Property	Applies to	Description
Concurrency	Topic subscribers Queue receivers	**Concurrency** on page 87
Delivery mode	Topic publishers Queue senders	**Delivery Mode** on page 91
Idle timeout	Topic publishers Queue senders	**Idle Timeout** on page 91
Maximum pool size	Topic publishers Queue senders	**Maximum Pool Size** on page 92

Property	Applies to	Description
Maximum wait time	Topic publishers Queue senders	**Maximum Wait Time** on page 92
Message selector	Topic subscribers Queue receivers	**Message Selector** on page 89
Priority	Topic publishers Queue senders	**Priority** on page 92
Steady pool size	Topic publishers Queue senders	**Steady Pool Size** on page 92
Transaction mode	Topic publishers Queue senders	**Transaction Mode** on page 93

7.1.3 Redelivery Handling Properties

Property	Applies to	Description
Delay	Topic subscribers Queue receivers	**Delay** on page 88
Move/Delete After N Times	Topic subscribers Queue receivers	**Move/Delete After N Times** on page 89
Action	Topic subscribers Queue receivers	**Action** on page 87
Move to Queue/ Topic	Topic subscribers Queue receivers	**Move to Queue/Topic** on page 90
Move to Destination Name	Topic subscribers Queue receivers	**Move to Destination Name** on page 90

7.1.4 Advanced Properties

Property	Applies to	Description
Durability	Topic subscribers	**Durability** on page 89
Server session batch size	Topic subscribers Queue receivers	**Server Session Batch Size** on page 90
Server session pool size	Topic subscribers Queue receivers	**Server Session Pool Size** on page 90

7.2 JMS Client Configuration Properties

The sections below describe the JMS client connector properties in detail., and supplement the information given in "Configuring JMS Clients" in the *Projects* chapter of the *Sun SeeBeyond eGate™ Integrator User's Guide*.

7.2.1 Consumers

Action

The **Action** property specifies the action to take following a specified number of unsuccessful attempts to deliver a message. This property specifies whether to **move** (redirect) or **delete** the message after the number of retries specified in **Delay** on page 88. See **Message Redelivery and Redirection** on page 28 for additional information.

Allowed Values

The allowed values are **move**, **delete**, and **no final action**.

Default Value

The default is **no final action**, which specifies continued retries until the message is received.

Concurrency

The **Concurrency** property specifies whether the message consumers use *connection consumer* or *serialized* processing. To use concurrent processing for a connection, select the **Connection consumer** setting. To use serial execution, select the **Serial mode** setting. This property applies to topic subscribers and queue receivers, and is specified in the *Basic* properties dialog.

JMS clients can be configured to use connection consumers to improve message throughput through concurrent processing. Connection consumers consume messages that are specified by a destination and an optional message selector (see **Message Selector** on page 89).

To start processing, a connection consumer gets a server session from its pool and loads the session with a message. Server sessions associate a JMS session with a thread. The server session pool is a set of server sessions provided to a connection consumer to process its messages.

The use of connection consumers increases message processing performance by enabling concurrent processing via multiple threads. You can specify the number of message driven beans (MDBs) or server session pool to assign to a JMS Collaboration to process messages concurrently. When you use connection consumer with fully concurrent or protected concurrent FIFO processing, this setting allows the integration server to assign multiple threads to execute the Collaboration on a particular message destination.

For queues, it is also possible to use connection consumer for concurrent processing on multiple CPUs (and application servers) on a system. This configuration does affect FIFO processing. For information, refer to **Message Processing Order** on page 22.

You specify the maximum number of threads per server session pool as described in **Server Session Pool Size** on page 90. By default, the maximum number of threads is **5**.

The maximum number of messages that a connection consumer can load into a server session at one time is set at **1** and cannot be changed (see **Server Session Batch Size** on page 90).

Default Value

The default JMS client concurrency mode is **Serial mode**.

Delay

The **Delay** property specifies the delay(s) to apply following a specified number of unsuccessful attempts to deliver a message. The format is **<retries>:<delay>**, where the number of retries is counted from the original rollback and the delay time is in milliseconds. Progressive delays can be specified by concatenating retry:delay pairs separated by a comma and a space:

```
<retry>:<delay>, <retry>:<delay>, …, <retry>:<delay>
```

This property applies to topic subscribers and queue receivers, and is specified in the *Redelivery Handling* properties dialog. See **Message Redelivery and Redirection** on page 28 for additional information.

Allowed Values

The maximum allowed delay is five seconds (**5000** ms).

Default Value

By default, no value is specified.

Durable Subscriber Name

The **Durable Subscriber Name** property, which is specified in the root *JMS Client* properties dialog, both provides a name for the JMS client and identifies it as being a durable subscriber.

A durable subscription is one that is not dependent upon a client's connection with a message server. Therefore, it is tolerant of disconnections, whether they are intentional or not. When a durable subscriber is disconnected from the message server, the server stores messages until the subscriber reconnects; it then delivers all accumulated messages that have not expired. This is also known as *store-and-forward* messaging.

Note: *Once set, the Durable Subscriber Name does not get autogenerated, and can only be changed manually. Copies of the Connectivity Map will also retain this name.*

Durability

The **Durability** property specifies whether or not the subscriber to this JMS connection is durable. When a subscriber is nondurable, the client sees messages on a topic only when the subscriber is active. If messages are published when the subscriber is inactive, the messages are lost. This property applies to topic subscribers only, and is specified in the *Basic* properties dialog.

When a subscriber is durable, messages are not lost even when the subscriber is inactive because the message server retains the messages until they are retrieved by the subscriber or until the messages expire. A durable subscriber registers with the message server as a durable subscriber with the name *source_destination*, for example, topicA_CollaborationA. When a subscriber becomes inactive, the message server retains the unexpired messages for a subsequent subscriber object with the same identity to resume the subscription. Note that there is a trade-off in performance.

Allowed Values

Durable or **Nondurable**.

Default Value

By default, JMS client connections are **Durable**.

Message Selector

The **Message selector** property specifies message selectors for the JMS client. This property applies to topic subscribers and queue receivers, and is specified in the *Basic* properties dialog.

To specify a message selector, enter a message selector String according to the JMS specification syntax. For example:

```
JMSType ='car' AND color ='blue'
```

If you use identifier start characters, it must be a character for which the **Character.isJavaIdentifierStart** method returns **true** as per JMS specification. If the identifier is invalid, the Java CAPS Monitor may show invalid message property names.

Move/Delete After N Times

The **Move/Delete After N Times** property specifies the number of retries to allow before redirecting or deleting the message, as specified in **Action** on page 87. The action is taken on the retry number entered for the property value.

This property applies to topic subscribers and queue receivers, and is specified in the *Redelivery Handling* properties dialog. See **Message Redelivery and Redirection** on page 28 for additional information.

Allowed Values

Any number.

Default Value

By default, no value is specified.

Move to Queue/Topic

The **Move to Queue/Topic** property specifies whether to redirect the message to a queue or a topic, following the number of retries specified in **Move/Delete After N Times** on page 89.

This property applies to topic subscribers and queue receivers, and is specified in the *Redelivery Handling* properties dialog. See **Message Redelivery and Redirection** on page 28 for additional information.

Allowed Values

The allowed values are **queue**, **topic**, and **auto** (which specifies the same kind of message destination as the message producer).

Default Value

By default, the destination is specified as **auto**.

Move to Destination Name

The **Move to Destination Name** property specifies a queue or topic name to which the message is to be redirected. The special character <$> specifies the original destination name.

This property applies to topic subscribers and queue receivers, and is specified in the *Redelivery Handling* properties dialog. See **Message Redelivery and Redirection** on page 28 for additional information.

Allowed Values

Any string.

Default Value

By default, no value is specified.

Server Session Batch Size

The **Server session batch size** property specifies the maximum number of messages that a connection consumer can load into a server session at one time. This property applies to topic subscribers and queue receivers, and is specified in the *Advanced* properties dialog. By default, this property is set to **1** and cannot be changed.

Server Session Pool Size

The **Server session pool size** property specifies the maximum number of threads per ServerSessionPool to be used for concurrent processing. This property applies to topic subscribers and queue receivers, and is specified in the *Advanced* properties dialog.

This property is used in conjunction with the connection consumer setting of the **Concurrency** property (**Concurrency** on page 87). You can specify the number of message driven beans (MDBs) or server session pool to assign to a JMS Collaboration to process messages concurrently. When you use connection consumer with fully concurrent or protected concurrent FIFO processing, this connection consumer

configuration allows the integration server to assign multiple threads to execute the Collaboration on a particular message destination.

For an overview about message processing, refer to **Message Processing Order** on page 22.

Allowed Values

An integer of **1** or larger, depending on the capability of the system, indicating the number of threads.

Default Value

By default, the maximum number of threads per server session pool is **5**.

7.2.2 Producers

Delivery Mode

The **Delivery mode** property specifies whether the messages for this JMS connection are persistent or non-persistent. This property applies to topic publishers and queue senders, and is specified in the *Basic* properties dialog.

Non-persistent delivery mode is the most efficient delivery mode; it does not require messages to be saved to permanent storage. Per JMS specification, the message destination delivers non-persistent messages with an at-most-once guarantee (the message is only delivered once, even if it is lost). There is a trade-off between performance and reliability; non-persistence offers better performance, but if a message server fails, non-persistent messages may be lost due to a power outage.

When messages are persistent, the message server places the message in permanent storage to ensure the message is not lost in transit if the message server fails. Persistent messages are delivered once, and only once.

For the JMS IQ Manager, persistent messages are stored in the message server database files.

Default Value

The default delivery mode is **Persistent**.

Idle Timeout

The **Idle timeout** property specifies the amount of time (in seconds) to wait before returning a connection to the pool. This property applies to topic publishers and queue senders, and is specified in the *Basic* properties dialog.

Default Value

The default timeout is **30** seconds.

Maximum Pool Size

The **Maximum pool size** property specifies the maximum number of connections to be made to the message server. This property applies to topic publishers and queue senders, and is specified in the *Basic* properties dialog.

Default Value

The default size is **32**.

Maximum Wait Time

The **Maximum wait time** property specifies the maximum amount of time (in milliseconds) to wait for acquiring a connection before throwing an exception. This property applies to topic publishers and queue senders, and is specified in the *Basic* properties dialog.

Default Value

The default time is **30000** milliseconds.

Priority

The **Priority** property specifies the message priority level for the JMS client. The message priority level that you specify causes all messages produced by this client to have that same priority level. For example, if you set the priority level to 2, all messages sent by that client have message priority level 2. This property applies to topic publishers and queue senders, and is specified in the *Basic* properties dialog.

You can also specify message priorities in Collaborations with the JMS OTD with the *setPriority* method. Collaboration message priorities override JMS client message priorities. For more information, refer to the *Sun SeeBeyond eGate™ Integrator User's Guide*.

Allowed Values

An integer between **0** and **9**, where 0 through 4 is normal priority and 5 through 9 is expedited priority.

Default Value

The default delivery mode is **4**.

Steady Pool Size

The **Steady pool size** property specifies the minimum (and initial) number of connections maintained in the pool. This property applies to topic publishers and queue senders, and is specified in the *Basic* properties dialog.

Default Value

The default is **4** connections.

Transaction Mode

The **Transaction mode** property specifies the transaction mode used for message producers (for consumers, it is always **XA**). This property applies to topic publishers and queue senders, and is specified in the *Basic* properties dialog.

The **Transaction mode** property specifies whether messages for this JMS client use one of the following transaction modes:

Transacted Mode

When you set the transaction mode to **Transacted**, the Java Collaboration Definition uses a new, separate transacted JMS session to send and receive messages. The message is committed automatically, and cannot be rolled back (even by raising an exception).

XA Mode

When the transaction mode is set to **XA**, the JMS session becomes part of the existing transaction, and is processed according to the XA two-phase commit protocol. In the first phase, the resource manager sends a query to commit to the receivers and waits for the receivers to respond with a confirmation. In the second phase, the resource manager receives confirmation from all receivers, and commits the message to all receivers. This setting prevents message loss and duplicate messages, even when a system unexpectedly shuts down.

Allowed Values

Transacted or **XA**.

Default Value

The default transaction mode is **XA**.

Note: *Documentation on distributed transaction processing using XA is available at no charge from The Open Group at* **http://www.opengroup.org** *(search on "XA").*

Chapter 8

JMS IQ Manager Runtime Configuration

JMS IQ Manager Runtime Configuration describes how to configure the runtime properties for JMS IQ Managers.

Note: *Configuring properties related to the generation of application files is described in the Sun SeeBeyond eGate™ Integrator User's Guide.*

What's in This Chapter

8.1 Accessing the Configuration Properties

The runtime configuration properties for JMS IQ Managers are accessed from the Enterprise Manager's *Integration Server Administration* application. Alternate procedures for opening this application are described in the *Sun SeeBeyond eGate™ Integrator System Administration Guide*. Once the application is open, clicking **Sun SeeBeyond JMS IQ Manager** in the explorer panel (see Figure 16) displays a set of configuration pages for the JMS IQ Manager, as shown in Figure 17.

Figure 16 Integration Server Administration Explorer Panel

Figure 17 Configuring Runtime JMS IQ Managers

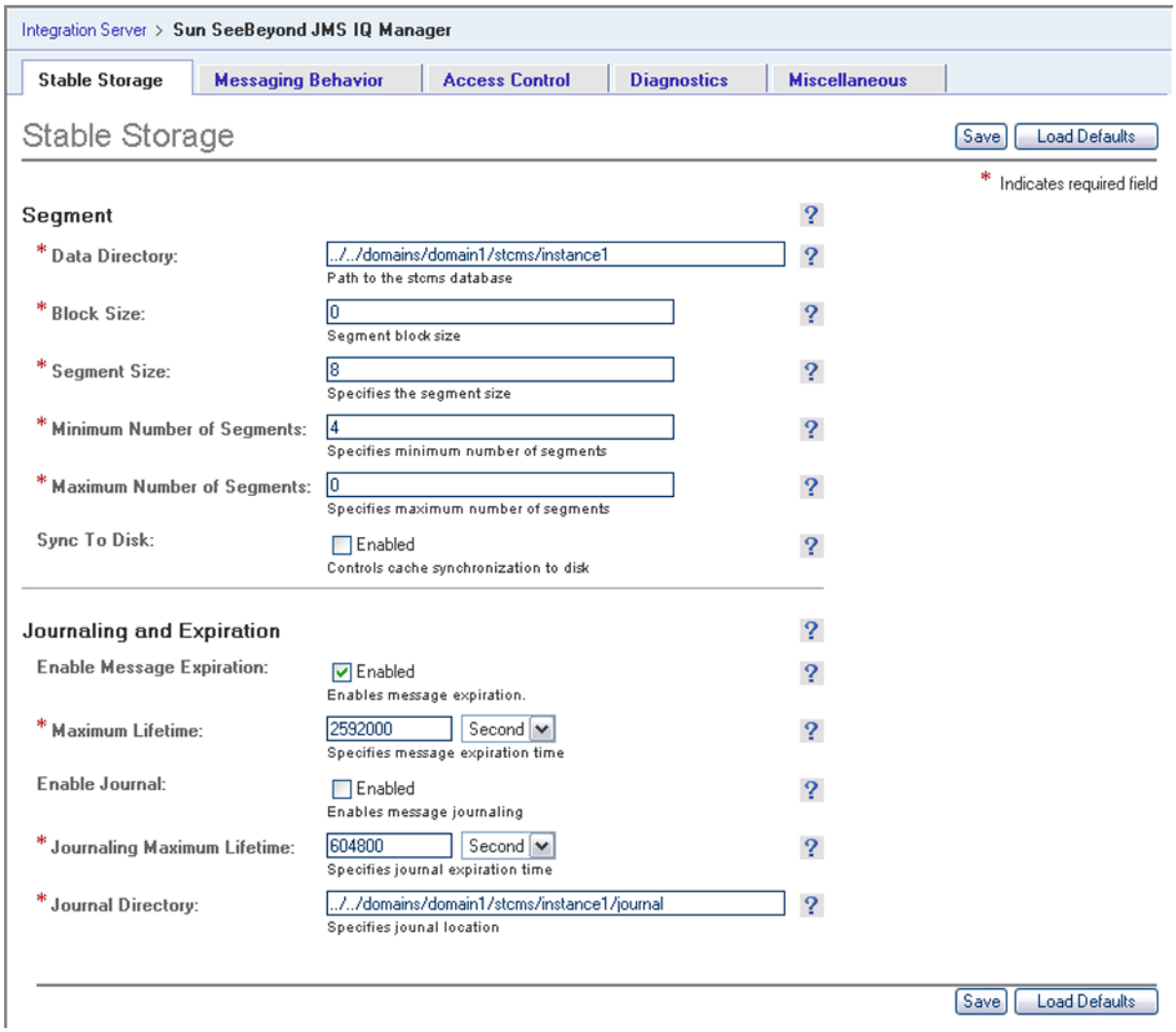


Table 14 Configuration Dialog Controls

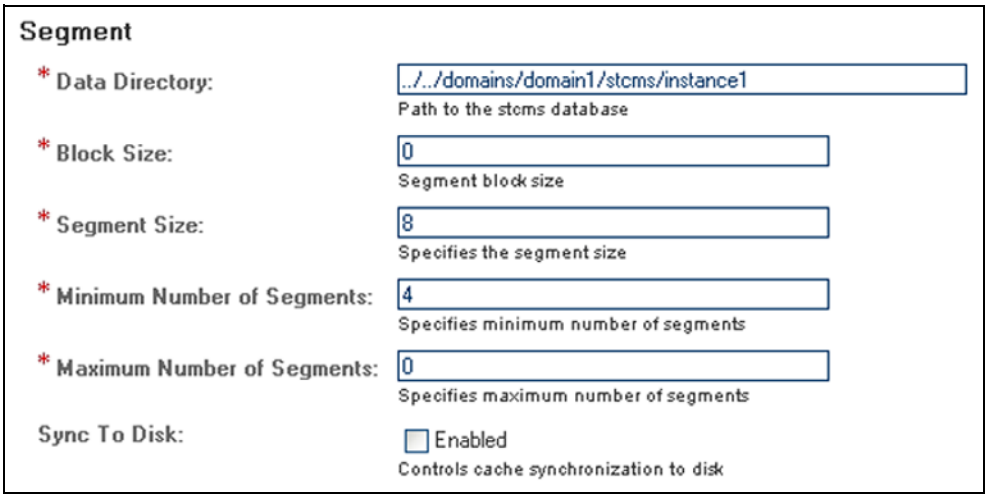
Button/Icon	Function
Save	Clicking this button saves your changes to the page. You must save your changes before proceeding to another page, or the changes will be lost.
Load Defaults	Clicking this button replaces all properties on the page with their default values.
?	Moving the cursor over this icon displays a description of the specific property or category.

8.2 Stable Storage Page

Note: *Modifying some properties, such as Block Size and Segment Size, requires changes to the database files. In such cases, you must manually delete the database files and restart the domain server before the changes take place. To prevent message loss, make sure there are no unread messages before deleting the database files.*

8.2.1 Segment Properties

Figure 18 Segment Properties Panel



As described in **JMS IQ Manager Database** on page 20, the JMS IQ Manager uses the JMS IQ Manager database to store persistent messages, and also messages that are larger than can be kept in the JMS IQ Manager memory, which is determined by the cache size setting (by default 0.5 MB for Windows and 1 MB for UNIX). You can specify several properties of this database file, as described in the following sections.

- **Data Directory** on page 96
- **Block Size** on page 97
- **Segment Size** on page 97
- **Minimum Number of Segments** on page 98
- **Maximum Number of Segments** on page 99

You can also specify whether or not the JMS IQ Manager controls the cache synchronization to disk by using the following option.

- **Sync to Disk** on page 99

Data Directory

The **Data Directory** property specifies where the JMS IQ Manager database files are located for JMS IQ Managers. You can specify the location as an absolute path, or as a

path relative to the **..\logicalhost\is\domains** directory. Using an absolute path for the data directory allows you the option of storing the JMS IQ Manager files on a different system, if desired.

If journaling is enabled, the data directory contains a **journal** directory, unless another location has been specified for the **Journal Directory** property. The journal directory holds the journaling database files. For information, refer to **Journal Directory** on page 101. Journaling is disabled by default.

Default location

The default setting is:

..\logicalhost\is\domains\domain_1\stcms\instance_1

Block Size

Data is read from and written to disk in units known as blocks. The **Block Size** property specifies the number of bytes per block.

Allowed Values

You can specify **0**, **512**, or **1024** bytes per block. If you specify **0**, the server will automatically determine the value by querying the operating system.

Default value

The default is **0** bytes per block.

Important: *Modifying the Block Size requires changes to the database files. You must manually delete the database files and restart the domain server before the changes take place. To prevent message loss, make sure there are no unread messages before deleting the database files.*

Segment Size

The JMS IQ Manager database consists of multiple memory-mapped database files known as segments. The **Segment Size** property specifies the total number of pages in each segment file. A page is 512 bytes on Windows, 1024 bytes on UNIX. The default segment size is 16,384 pages, which is 8 MB for Windows and 16 MB for UNIX. By default, these segments are named **stcms*.dbs** and reside in the message server folder on the Logical Host.

You should set the segment size to a value larger than the sum of:

- The anticipated number of subscribers.
- The anticipated maximum transaction size in bytes divided by the page size in bytes.

  (The transaction size is the sum of the sizes of all messages in one transaction. If transactions span no more than one message, the maximum transaction size is equal to the size of the largest message.)
- An additional ten pages to allow for overhead.

For example, on a UNIX system (where the page size is 1 KB) where you expect no more than 100 subscribers and that messages will not exceed 100 KB, and that only one message will be sent/received per transaction, you would set the segment size to at least 100 + (100kb/1kb) + 10 = 210 pages

With this setting, there may only be one (100,000 byte) message in each segment. The ideal segment size depends on the circumstances. If the slowest subscriber lags behind the fastest publisher by a certain number of messages, you can set the segment size so that this number of messages will fit a single segment.

The JMS IQ Manager cleans up the database by recycling segments for which all messages have either expired or have been retrieved by their subscribers.

A lower segment size setting results in more efficient use of the disk because smaller segments turn over more rapidly and thus provide more effective use of server memory. However, a lower segment size means that more new segments may need to be allocated, which requires more time than freeing a cleaned-up segment. In addition, if a transaction is larger than the specified segment size, the server rolls back the transaction. You must then increase the **Segment Size** property to an amount larger than the message.

A high segment size setting can be advantageous in that cleanup runs less often; but each cleanup takes longer. However, cleaning up two small segments requires more time than cleaning up one large segment, so you can set a large segment size to increase performance on systems that are constrained by disk I/O speed rather than memory or space.

Allowed Values

An integer greater than **1**. Set this property to at least twice the total number of anticipated durable subscribers.

Default value

The default segment size is **16,384** pages, which is **8 MB** for Windows and **16 MB** for UNIX.

Important: *Modifying the Segment Size requires changes to the database files. You must manually delete the database files and restart the domain server before the changes take place. To prevent message loss, make sure there are no unread messages before deleting the database files.*

Minimum Number of Segments

The **Minimum Number of Segments** property specifies the minimum number of database files (segments) that the JMS IQ Manager creates initially for stable message storage. When the minimum is exceeded, the server allocates additional segments on an as-needed basis, up to the number of files specified for the **Maximum Number of Segments** property as described in **Maximum Number of Segments** on page 99.

In addition to limiting the maximum number of segments, you can also specify the size limit for segments. For more information about the **Segment Size** property, refer to **Segment Size** on page 97.

Allowed Values

An integer from **1** through **99,999** indicating the number of segments.

Default value

The default is **4** segments.

Maximum Number of Segments

The **Maximum Number of Segments** property specifies the upper limit for the number of database files (segments) that the JMS IQ Manager creates for its stable message storage. You use this property to limit the amount of disk space that the JMS IQ Manager uses. If the JMS IQ Manager attempts to write data that exceeds this limit, it exits gracefully and logs an error message in the JMS IQ Manager log.

Allowed Values

An integer from **0** through **99,999** indicating the number of segments.

Default value

The default is **0**. This value causes the JMS IQ Manager to create new files as needed, limited only by available disk space.

Sync to Disk

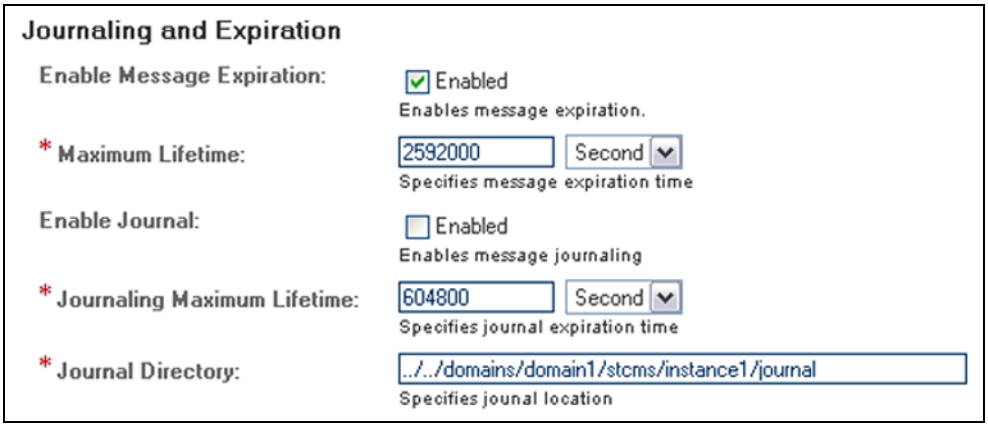
The **Sync to Disk** property specifies whether the JMS IQ Manager controls cache synchronization to disk. When you disable cache control, the operating system controls the synchronization schedule. Disabling cache control increases performance, but also increases risk of message loss in the event of system failure.

Default condition

The **Sync to Disk** property is *disabled* by default.

8.2.2 Journaling and Expiration Properties

Figure 19 Journaling and Expiration Properties Panel



Journaling messages allows you to re-publish messages at a later date. You can specify several options and properties for journaling, as described in the following sections.

- **Enable Message Expiration** on page 100
- **Maximum Lifetime** on page 100
- **Enable Journal** on page 100
- **Journaling Maximum Lifetime** on page 101
- **Journal Directory** on page 101

To re-publish journaled messages, you use the STC MS Control utility as described in **Republishing Messages from Topics** on page 117 or **Republishing Messages from Queues** on page 117. You can also use the STC MS Control utility to browse journaled messages with the **-journaler** flag. For information, refer to **Browsing Journaled Messages** on page 117.

Enable Message Expiration

The **Enable Message Expiration** option allows you to enable or disable message expiration for JMS IQ Managers. When you enable message expiration, messages are removed from the queue after the time specified for the **Maximum Lifetime** property has expired.

Maximum Lifetime

The **Maximum Lifetime** property specifies the maximum amount of time before a live message expires. After it expires, the message is removed from the queue whether it has been consumed or not. If you specify **0**, the message never expires.

Default value

The default is **2592000** seconds (30 days).

Enable Journal

The **Enable Journal** option allows you to enable or disable journaling for JMS IQ Managers. When you enable journaling, every inbound message is automatically copied to the journal database. The message is then held in the journal database for the duration of time to live specified for journaled messages.

By default, the expiration time for a journaled message is 7 days. To change the time to live for journaled messages, refer to **Journaling Maximum Lifetime** on page 101. The journaled message time to live is completely independent of when the live counterpart of the message is consumed by its publisher.

When a journaled message expires, it is not deleted from the journal database — it remains there until you back up the topics or queues. When you back up, all messages in the journal database are included in the archive, and the journal expired messages are removed from the journal database. It is recommended that you back up daily when journaling is enabled; otherwise, the journal database retains journal expired messages and may grow exceedingly large. Because the journal database and the JMS

IQ Manager database are located on the same system, it is important to avoid running out of disk space.

To back up the journal database, you use the MS Control Utility. For more information, refer to **Backing Up** on page 119.

When messages are in the journal database, you can view them but not edit them. You can use either Enterprise Manager or the STC MS Control utility to view and republish journaled messages. For more information about the MS Control utility, see **JMS IQ Manager Management** on page 110.

Default condition

Journaling is *disabled* by default.

Journaling Maximum Lifetime

The **Journaling Maximum Lifetime** property specifies the maximum amount of time that a journaled message persists before it expires. The JMS IQ Manager journals messages only when journaling is enabled as described in **Enable Journal** on page 100. Journaling is disabled by default.

When a journaled message expires, it is not deleted from the journal database — it remains there until you back up the topics or queues. When you back up, all messages in the journal database are included in the archive (.zip file), and the journal expired messages are removed from the journal database.

Default value

The default is **604800** seconds (7 days).

Journal Directory

The **Journal** directory holds the journal database files and the journaling log file. You can enter an absolute path or a path relative to the **..\logicalhost\is\domains** directory. Using an absolute path for the data directory allows you to store the journal database files on a different system; for example, for backup purposes.

The JMS IQ Manager only creates a journal directory when journaling is enabled. Journaling is disabled by default. For more information, refer to **Enable Journal** on page 100.

Allowed Values

An absolute path or a path relative to the **..\logicalhost\is\domains** directory.

Default value

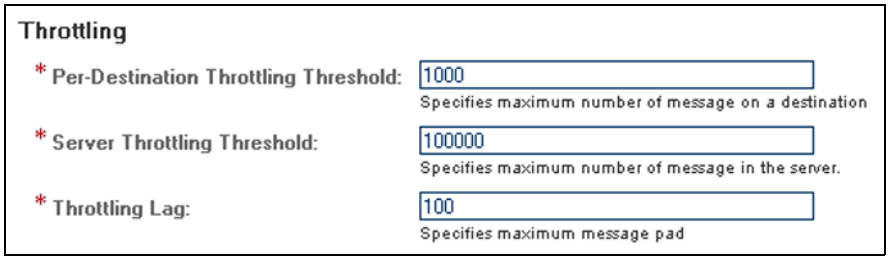
By default, the journal database files are stored in the following folder:

..\logicalhost\is\domains\domain_1\stcms\instance_1\journal

8.3 Messaging Behavior Page

8.3.1 Throttling Properties

Figure 20 Throttling Properties Panel



As described in **Performance Issues** on page 34, you can use the JMS IQ Manager as a semi-permanent storage medium only if you have sufficient memory and disk resources. To manage the memory and disk resources needed by the JMS IQ Manager, you use the publisher throttling feature. You can specify several properties for throttling, as described in the following sections.

- **Per-Destination Throttling Threshold** on page 102
- **Server Throttling Threshold** on page 102
- **Throttling Lag** on page 103

Per-Destination Throttling Threshold

The **Per-Destination Throttling Threshold** property specifies the maximum number of messages per topic or queue after which all producers of the message destination are throttled, assuming the **Server Throttling Threshold** has been exceeded. Once a producer is throttled, the JMS IQ Manager stops reading messages from it until the number of undelivered messages in the affected destination has dropped to less than the difference between the value specified by the **Per-Destination Throttling Threshold** and the value specified by the **Throttling Lag**. See also **Throttling Lag** on page 103.

Allowed Values

An integer from **0** through **999,999,999** indicating the number of messages. If you specify **0**, the publishers are never throttled.

Default value

The default is **1000** messages.

Server Throttling Threshold

The **Server Throttling Threshold** property specifies the total number of messages for all message destinations combined before the JMS IQ Manager starts throttling any

producers. For a detailed explanation and an example, see **Throttling Producers** on page 34.

Allowed Values

An integer from **0** through **999,999,999** indicating the number of messages. If you specify **0**, no producers are ever throttled.

Default value

The default is **100,000** messages.

Throttling Lag

You use the **Throttling Lag** property in combination with the **Per-Destination Throttling Threshold** property. Once a producer is throttled, the JMS IQ Manager stops reading messages from it until the number of undelivered messages in the affected destination has dropped to less than the difference between the value specified by the **Per-Destination Throttling Threshold** and the value specified by the **Throttling Lag**. See also **Per-Destination Throttling Threshold** on page 102.

Allowed Values

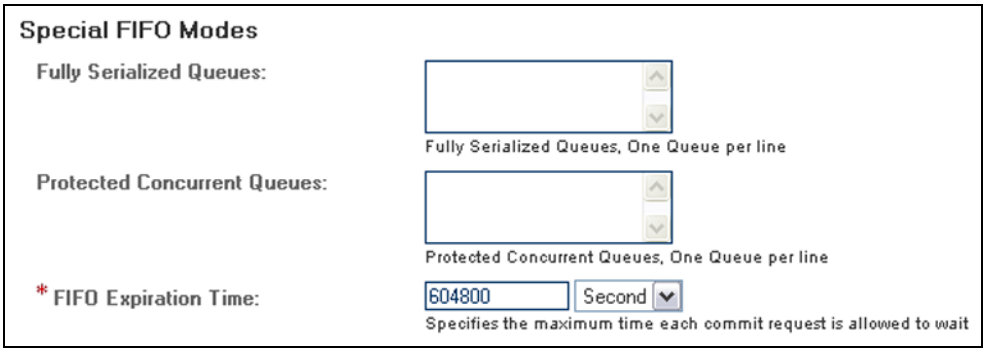
An integer from **0** through **99,999,999**. The value must be less than that of the **Per-Destination Throttling Threshold** property.

Default value

The default is **100** messages.

8.3.2 Special FIFO Mode Properties

Figure 21 Special FIFO Modes Properties Panel



As described in **Message Delivery Order** on page 18, the JMS IQ Manager allows three different FIFO delivery modes: fully concurrent, protected concurrent, or fully serialized. By default, all message destinations use the *fully concurrent* delivery mode (see **Fully Concurrent Processing** on page 23).

In fully concurrent mode, receivers can retrieve messages from a destination only when all other messages have been received or are in the process of being received. Receivers can then commit messages without restrictions. This means that the messages can be committed out of sequence, which is not always desirable.

In the **Special FIFO Modes** panel, you can specify alternate properties for delivery order, as described in the following sections.

- **Fully Serialized Queues** on page 104
- **Protected Concurrent Queues** on page 104
- **FIFO Expiration Time** on page 104

Fully Serialized Queues

The **Fully Serialized Queues** property specifies which queues are fully serialized. In fully serialized mode, receivers can only retrieve messages after all previous messages for the message destination have been received *and* committed.

To implement serialized mode across multiple application servers, you must set the JMS clients for the consumers involved to serial mode. For more information, refer to **Concurrency** on page 87.

Allowed Values

A list of existing queue names.

Protected Concurrent Queues

The **Fully Concurrent Queues** property specifies which queues use protected concurrent FIFO delivery mode. In protected concurrent mode, a receiver can retrieve messages just as in fully concurrent mode (only after all messages have been received or are being received), but messages can only be committed if all previous messages have been committed.

Allowed Values

A list of existing queue names.

FIFO Expiration Time

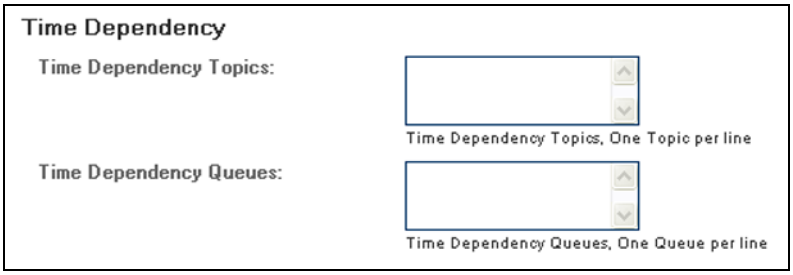
The **FIFO Expiration Time** property specifies the maximum amount of time to delay a *commit* request in special FIFO mode.

Default value

The default FIFO mode expiration time is **604800** seconds (168 hours).

8.3.3 Time Dependency Properties

Figure 22 Time Dependency Properties Panel



By default, messages are processed by and delivered to Collaborations in the order in which they were committed to their destination, independent of messages associated with any other destination. Setting a specific time dependency causes the message processing order to be dependent on messages associated with *other* destinations. These destinations are specified using the **Time Dependency Topics** and **Time Dependency Queues** properties.

Messages associated with any of the destinations in the time dependency group are ordered in fully serialized mode. In other words, a message associated with a destination in this group is processed only after all older messages associated with any other destination in the time dependency group have been processed.

Note: *Message properties such as JMS **priority** have no effect when time dependency is used.*

For a general overview of message processing order, refer to **Message Delivery Order** on page 18.

Allowed Values

A list of queue or topic names.

If you specify a message destination that does not exist, Enterprise Manager enables time-based order for all other destinations and ignores the unknown name. This allows you to add topics and queues in your project at a later time.

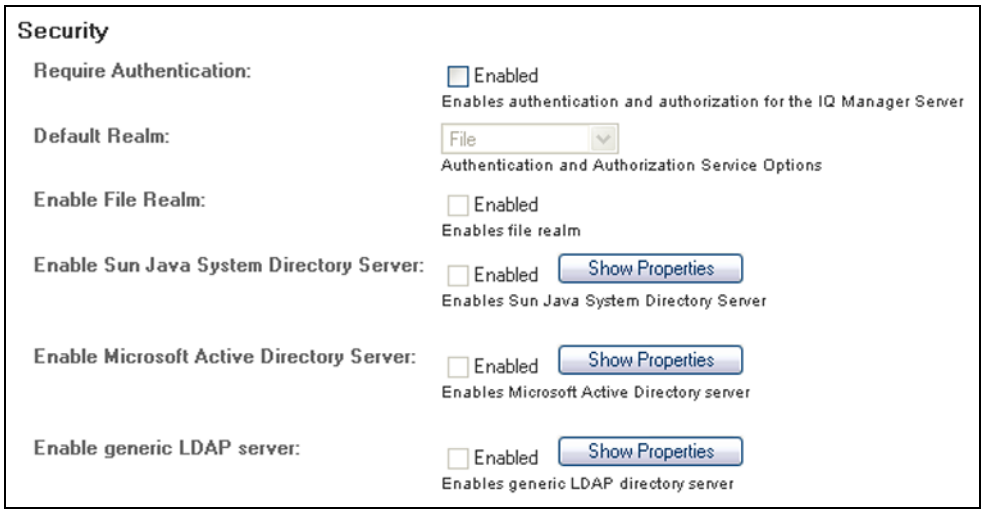
Note: *When you specify time dependency, you cannot use a colon (:) or semicolon (;) in topic or queue names because they are already used for the time dependency value.*

8.4 Access Control Page

Options on the **Access Control** page allow you to enable password authentication for the JMS IQ Manager and supported LDAP servers. Use of this page is described in the *Sun SeeBeyond eGate™ Integrator System Administration Guide*.

Security Options

Figure 23 Security Properties Panel



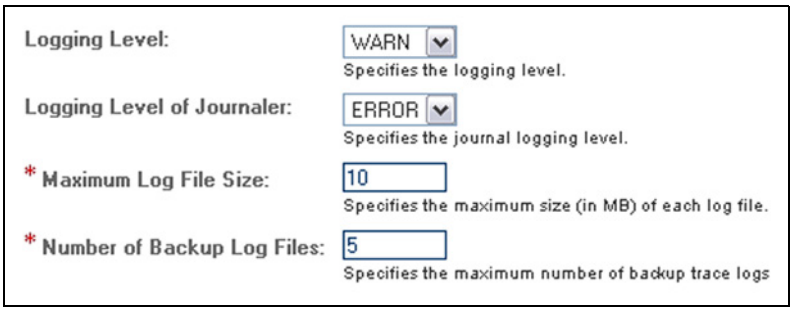
Default condition

JMS IQ Manager security is disabled by default.

8.5 Diagnostics Page

8.5.1 Diagnostic Properties

Figure 24 Diagnostic Properties Panel



The generation of log files for diagnostic purposes can have a major impact on system resources. You can specify several properties for logging, as described in the following sections.

- **Logging Level** on page 107
- **Logging Level of Journaler** on page 108
- **Maximum Log File Size** on page 108
- **Number of Backup Log Files** on page 108

Logging Level

The **Logging Level** property specifies the threshold severity level at which the system issues log messages. The JMS IQ Manager will only issue log messages having a severity level that is higher than or equal to the specified level.

Table 15 Logging Levels

Level	Message Types Issued
FATAL	Fatal
ERROR	Fatal, Error
WARN	Fatal, Error, Warning
INFO	All

Default

By default, the logging level is specified as **WARN**, and the JMS IQ Manager issues warning, error, and fatal messages.

Logging Level of Journaler

The **Logging Level of Journaler** property specifies the threshold severity level at which the system journals log messages. The JMS IQ Manager will only journal log messages having a severity level that is higher than or equal to the specified level.

Table 16 Journaler Logging Levels

Level	Message Types Journaled
FATAL	Fatal
ERROR	Fatal, Error
WARN	Fatal, Error, Warning
INFO	All

By default, the journal log file resides in the **journal** directory in the JMS IQ Manager directory. The location of this directory can be specified with the **Journal Directory** property as described in **Journal Directory** on page 101.

Default condition

By default, the journaling level is specified as **ERROR**, and the JMS IQ Manager journal log includes only error and fatal messages.

Maximum Log File Size

You can specify the maximum size for the JMS IQ Manager log file with the **Maximum Log File Size** property. If the JMS IQ Manager attempts to log more than the specified log file size, the log file is renamed to **stcms.log.<*N*>**, and a new file is created.

The variable ***N*** is a number between 1 and the number specified by the **Number of Backup Log Files** property. By default, the JMS IQ Manager can create five backup log files.

Allowed Values

An integer larger than **0**, indicating the size of the log file in MB.

Default value

The default log file size is **10** MB.

Number of Backup Log Files

You can specify the maximum number of backup log files in the JMS IQ Manager database with the **Number of Backup Log Files** property. When the specified number is exceeded, the oldest log file is discarded.

Allowed Values

An integer greater than **0**, indicating the number of backup log files.

Default value

The default number of backup log files is **5**.

8.6 Miscellaneous Page

8.6.1 Enable Alert Option

Figure 25 Enable Alert Option Panel

When the **Enable Alert** option is enabled, the JMS IQ Manager will generate alerts when particular events occur, such as reaching a throttling threshold.

Default condition

Alert generation is *disabled* by default.

Chapter 9

JMS IQ Manager Management

You can manage JMS IQ Managers either from Enterprise Manager (see the *Sun SeeBeyond eGate™ Integrator System Administration Guide*) or the command-line MS Control utility. *JMS IQ Manager Management* describes the use of the MS Control utility for managing JMS IQ Managers.

What's in This Chapter

9.1 Overview of MS Control Utility Features

The MS Control utility is a command-line utility that enables you to manage many advanced aspects of the JMS IQ Managers. With the MS Control utility you can manage JMS IQ Managers as follows:

- **Message Servers**
  - Display the version of the JMS IQ Manager.
  - Shut down the JMS IQ Manager.
- **Message Destinations**
  - For a specified JMS IQ Manager: List, create, or delete topics or queues.
  - For a specified topic: List, create, or delete subscribers; retrieve a message list; view statistics.
  - For a specified queue: List, create, or delete receivers; retrieve a message list; view queue statistics.
  - For a specified queue or topic: Create, delete, modify, monitor, or list the contents of the queue or topic.
- **Messages**
  - For a specified message: View, delete, or modify message content.
  - View or modify a particular message type.
  - Fetch or delete a range of messages.

- Journal, back up, and archive messages.

9.2 Flags and Arguments

Table 17 MS Control Utility Flags and Arguments

Shortcut	Flag *arguments*	Purpose
-ar	**-archiver** *directoryname*	Browse specified archive. See **Browsing Archives** on page 119.
-b	**-backup** *file date*	Back up messages. See **Backing Up** on page 119.
-cqm	**-changeqmsg** queuename *seqnumber*	Change the content of the message at the specified sequence number in the specified queue, reading from standard input (the command prompt, or whatever file or piped command it specifies).
-ctm	**-changetmsg** *topicname seqnumber*	Change the content of the message at the specified sequence number in the specified topic, reading from standard input (the command prompt, or whatever file or piped command it specifies). See **Changing Topic Message Contents** on page 114.
-cq	**-createqueue** *queuename*	Create a new queue with the specified name.
-cs	**-createsub** *topicname subname clientname*	Create a new subscriber for the specified topic and client. For *clientname*, specify **eGate**.
-ct	**-createtopic** *topicname*	Create a new topic with the specified name.
-dq	**-deletequeue** *queuename*	Delete the specified queue.
-ds	**-deletesub** *topicname subname clientname*	Delete a certain subscriber from the specified topic and client. For *clientname*, specify **eGate**.
-dt	**-deletetopic** *topicname*	Delete the specified topic.
-dqm	**-delqmsg** *queuename*	Delete the message at the specified sequence number in the specified queue.
-dqm	**-delqmsg** *queuename seqnumber*	Delete the message at the specified sequence number in the specified queue.
-dtm	**-deltmsg** *topicname seqnumber*	Delete the message at the specified sequence number in the specified topic.
	--help	View help information.
	-host *hostname*	Specify the name of the Logical Host. If not specified, the default is: **-host localhost** For hosts other than localhost and flags other than --help and --version, **-host** is *required*.
-j	**-journaler**	Browse journaled messages. See **Browsing Journaled Messages** on page 117.

Table 17 MS Control Utility Flags and Arguments (Continued)

Shortcut	Flag *arguments*	Purpose
-lt	**-locktopic** *topicname*	Lock a topic from being accessed, prevent any subscriber from receiving messages from it.
	-msgtype *type*	Specify the data type of the content of the message. Must be **bytes** or **text**.
	-msversion	View server version information.
	-offset *portoffset*	Specify a server port offset number.
	-port *portnumber*	Specify the TCP/IP port of the Logical Host that this Message Server is listening to. If not specified, the default is: **-port 7555** For ports other than 7555 and flags other than --help and --version, **-port** is *required*.
-gqm	**-qmessage** *queuename*	Retrieve the particular message designated by *seqnumber* for the specified queue. If the specified queue contains no message with this sequence number, an error is returned.
-qmi	**-qmimport** *topicname seqno nmgs*	Republishing messages from a queue. See **Republishing Messages from Queues** on page 117.
-qml	**-qmsglist** *queuename seqnumber numbermessages*	List all messages for the specified queue, starting at or above the specified sequence number, and listing no more than *numbermessages* altogether.
-ql	**-queuelist**	List all queues for this server. See **Viewing All Queues for a JMS IQ Manager** on page 116.
-qs	**-queuestat** *queuename*	View statistics for a specific queue. See **Displaying the Status of Queues** on page 116.
-rla	**-recvlistall**	List all receivers for all queues combined. See **Viewing Properties of All Receivers** on page 116.
-rlfq	**-recvlistforqueue** *queuename*	List all receivers for the specified queue. See **Viewing Properties of All Receivers of Queues** on page 117.
	-shutdown	Shut down the server. See **Shutting Down the Server** on page 114.
	-status	View server status. See **Viewing JMS IQ Manager Status** on page 114.
-sla	**-sublistall**	List all subscribers for all topics combined. See **Viewing Properties of All Subscribers** on page 115.
-slft	**-sublistfortopic** *topicname*	List all subscribers for the specified topic. See **Viewing Properties of All Subscribers to Topics** on page 116.

Table 17 MS Control Utility Flags and Arguments (Continued)

Shortcut	Flag *arguments*	Purpose
-timeout	**-timeout** *seconds*	Specify the timeout in seconds. See **Setting Timeout** on page 120.
-gtm	**-tmessage** *topicname seqnumber*	Retrieve the particular message designated by *seqnumber* for the specified topic. If the specified topic contains no message with this sequence number, an error is returned.
-tmi	**-tmimport** *topicname seqno nmgs*	Republish messages from a topic. See **Republishing Messages from Topics** on page 117.
-tml	**-tmsglist** *topicname seqnumber numbermessages*	List all messages for the specified topic, starting at or above the specified sequence number, and listing no more than *numbermessages* altogether.
-tl	**-topiclist**	List all topics for this server. See **Viewing All Topics for a JMS IQ Manager** on page 114.
-ts	**-topicstat** *topicname*	View statistics for the specified topic. See **Viewing the Status of Topics** on page 115.
-ut	**-unlocktopic** *topicname*	Unlock a topic, restoring access to all subscribers.
	-username *username*	Supply the user name to connect to the JMS IQ Manager. This flag is mandatory when JMS IQ security is enabled.
	-**userpassword** *userpassword*	Supply the password to connect to the JMS IQ Manager. This flag is mandatory when JMS IQ security is enabled.
	--version	View utility version information.

9.2.1 Syntax

The MS Control utility uses the following syntax for all flags other than **--help** and **--version**:

```
stcmsctrlutil -host hostname -port portnumber [-offset portoffset]
              -flag

stcmsctrlutil -host hostname -port portnumber [-offset portoffset]
              -flag argument1 [argument2 [argument3]]
```

If JMS IQ Manager security is enabled, you must specify a user name and password as follows:

```
stcmsctrlutil -host hostname -port portnumber -username username
              - userpassword userpassword [-offset portoffset] -flag
```

For more information about security, refer to *Sun SeeBeyond eGate™ Integrator System Administration Guide*.

For **--help** and **--version**, the syntax is as follows:

```
stcmsctrlutil --help

stcmsctrlutil --version
```

9.3 Using the MS Control Utility

9.3.1 Shutting Down the Server

```
stcmsctrlutil -host localhost -port 24055 -shutdown
```

9.3.2 Viewing JMS IQ Manager Status

```
stcmsctrlutil -host localhost -port 24055 -status
Up since: Tue Oct 14 20:54:23 2003
Memory used by data messages: 950.729 K(Bytes)
Total messages passed through: 1900331
Total messages retained: 3555
Number of message queue(s): 9
Number of connection(s): 14
Port number: 18007
Process ID: 2780
Server state: Ready and running...
```

9.3.3 Viewing All Topics for a JMS IQ Manager

```
stcmsctrlutil -host localhost -port 24055 -topiclist
Topic List:
    SeeBeyond.MS.Control
    Broadcast
    STCTemporaryTopic.2.1
```

9.3.4 Changing Topic Message Contents

To change the contents (payload) of a message for a specified topic, you use the **-changetmsg** flag. You must specify whether the message type is bytes or text. The message cannot be processed while you are changing the contents.

To change topic message contents

1. Use the following command syntax to specify the contents change:

   ```
   stcmsctrlutil.exe -p portnumber -ctm topicname seqnumber -msgtype
   type
   ```

   where *portnumber* is the number of the port, where *topicname* is the name of the topic that contains the message, where *seqnumber* is the sequence number of the message, and where *type* is either bytes or text indicating the message type of the original message.

2. Press **ENTER**.
3. Type the new contents.

4 Press **CTRL-Z**.

The following example illustrates changing the contents of a message on topic T0:

```
stcmsctrlutil -ctm T0 182 -p 18007 -msgtype text
NEWCONTENTS
^Z
Message: 182 has been changed
```

The following shows the revised contents of the message:

```
stcmsctrlutil.exe -p 18007 -tmessage T0 182 -msgtype text
NEWCONTENTS
```

9.3.5 Viewing the Status of Topics

The following example displays the current status of the topic **Broadcast**:

```
stcmsctrlutil -host localhost -port 18007 -topicstat Broadcast
Topic Name: Broadcast
First enqueue time: 05172001:16:30:30
Last enqueue time: 05172001:16:30:42
Number of current subscribers: 0
Number of total subscribers: 2
Message count: 6
Min Sequence Number: 0
Max Sequence Number: 3
Suspended: No
```

The **Suspended** entry shows whether topic is suspended and all subscribers stop receiving messages. This usually shows **No**. After the JMS IQ Manager restarts, all topics show **Suspended: No status**.

The enqueue times and sequence numbers are described in **Enqueued Message Properties** on page 32.

9.3.6 Viewing Properties of All Subscribers

```
stcmsctrlutil -host localhost -port 24055 -sublistall
Number Of Subscriber(s): 4
Subscriber name: NonDurable1
        Client ID:
        Topic name: SeeBeyond.MS.Control
        Committed sequence: 0
        High sequence: 0
Subscriber name: subscriber1
        Client ID: Client
        Topic name: Broadcast
        Committed sequence: 0
        High sequence: 3
Subscriber name: subscriber2
        Client ID: Client
        Topic name: Broadcast
        Committed sequence: 3
        High sequence: 6
Subscriber name: NonDurable2
        Client ID:
        Topic name: STCTemporaryTopic.2.1
        Committed sequence: 0
        High sequence: 0
```

9.3.7 Viewing Properties of All Subscribers to Topics

```
stcmsctrlutil -host localhost -port 24055 -sublistfortopic STC
Number Of Subscriber(s): 2
Subscriber name: subscriber1
        Client ID: Client
        Topic name: STC
        Committed sequence: 0
        High sequence: 3
Subscriber name: subscriber2
        Client ID: Client
        Topic name: STC
        Committed sequence: 3
        High sequence: 6
```

9.3.8 Viewing All Queues for a JMS IQ Manager

```
stcmsctrlutil -host localhost -port 24055 -queuelist
Queue List:
    MyQueue0
    PTP
```

9.3.9 Displaying the Status of Queues

The following example displays the current status of the queue **PTP**:

```
stcmsctrlutil -host localhost -port 24055 -queuestat PTP
Queue Name: PTP
First enqueue time: 02011970:00:00:00
Last enqueue time: 02011970:00:00:00
Number of current receivers: 2
Message count: 4
Messages sent and committed: 245
Min sequence number: 245
Max sequence number: 248
Suspended: No
```

The enqueue times and sequence numbers are described in **Enqueued Message Properties** on page 32.

9.3.10 Viewing Properties of All Receivers

```
stcmsctrlutil -host localhost -port 24055 -recvlistall
Number Of Receiver(s): 3
Receiver ID: 14235659
        Queue name: MyQueue0
        Session ID: 1
        Committed messages: 0
        Uncommitted messages: 0
Receiver ID: 14274653
        Queue name: PTP
        Session ID: 3
        Committed messages: 434
        Uncommited messages: 0
Receiver ID: 14291939
        Queue name: PTP
        Session ID: 4
        Committed messages: 432
        Uncommited messages: 1
```

9.3.11 Viewing Properties of All Receivers of Queues

```
stcmsctrlutil -host localhost -port 24055 -recvlistforqueue PTP
Number Of Receiver(s): 2
Receiver ID: 14274653
        Queue name: PTP
        Session ID: 3
        Committed messages: 434
        Uncommited messages: 0
Receiver ID: 14291939
        Queue name: PTP
        Session ID: 4
        Committed messages: 432
        Uncommitted messages: 1
```

9.3.12 Republishing Messages from Topics

To republish journaled messages from topics, you use the **-tmimport** flag.

The following example republishes five journaled messages from topic T0 starting from message with sequence number 491.

```
stcmsctrlutil -j -tmi T0 491 5
Executed function: IMPORT
Importing messages
Last imported sequence number = 491
Last imported sequence number = 497
```

9.3.13 Republishing Messages from Queues

To republish journaled messages from queues, use the **-qmimport** flag.

The following example republishes five journaled messages from queue T0 starting from message with sequence number 500.

```
stcmsctrlutil -j -qmi T0 500 5
Executed function: IMPORT
Importing messages
Last imported sequence number = 500
Import failed
Import failed on sequence number: 500
```

This example will fail because there are no messages. Import failed trying to republish first sequence number. To republish messages from archive you must specify the **-ar** flag and archive directory instead of the **-journaler** flag.

9.3.14 Browsing Journaled Messages

To browse journaled messages, you use the **-journaler** flag. The **-journaler** flag receives information from the journaler instead of the JMS IQ Manager. The journaler does not support information about subscribers and receivers; flags such as **-sublistall, -deletesub,** or **-recvlistall** do not work with the **-journaler** flag. You cannot delete journaled messages, topics, or queues. The MS Control utility displays information about journaled topics and queues in the same format as the JMS IQ Manager.

The following example displays the topic message list from the JMS IQ Manager:

```
stcmsctrlutil -tl
```

```
Number Of Topic(s): 4
Topic List:
        STCMS.Control
        STCMS.Journal
        T0
        STCTemporaryTopic.1031789365648.1031789335025.1
```

The following example displays the topic message list from the journaler:

```
stcmsctrlutil -j -tl
Number Of Topic(s): 1
Topic List:
        T0
```

The following example displays information about queue Q0 from the JMS IQ Manager:

```
stcmsctrlutil -qs Q0
Queue Name: Q0
First enqueue time: 01011970:00:00:00
Last enqueue time: 01011970:00:00:00
Number of current receivers: 0
Message count: 0
Messages sent and committed: 1001
Min sequence Number: 0
Max sequence Number: 0
```

The following example displays the same information about queue Q0, but from the journaler:

```
stcmsctrlutil -j -qs Q0
Queue Name: Q0
First enqueue time: 09122003:00:14:07
Last enqueue time: 09122003:00:14:28
Number of current receivers: 0
Message count: 1001
Messages sent and committed: 0
Min sequence Number: 0
Max sequence Number: 1000
```

The following example displays the information about one message with sequence number 0. Because the message has been consumed, the MS Control utility cannot display this information.

```
stcmsctrlutil -qml Q0 0 1
Number Of Messages(s): 0
```

The following example displays the same information as above, but from the journaler. The message is not journal expired, which enables the MS Control utility to display the message properties.

```
stcmsctrlutil -j -qml Q0 0 1
Number Of Messages(s): 1
Message[1]:
Message.SeqNo=0
Message.Timestamp=1031789647260
Journaler.ExpirationTime=1031809647260
Message.Size=228
Message.JMSProperty.TS=1031789647260
Message.JMSProperty.EX=0
Message.JMSProperty.DM=1
Message.JMSProperty.TY=ASCII
Message.JMSProperty.PR=0
Message.JMSProperty.RD=false
Message.JMSProperty.MI=ID:377:3b742aa5:950:0a01beee:3d7fdc4f104
```

```
Message.UserProperty.JMS_ProducerID=BENCH
```

9.3.15 Backing Up

The **-backup** flag creates a zip file that contains all messages (regardless of whether they are live or journaled) for all queues and topics up to the specified date. After you create this zip file, you can unzip it and then browse the archive using the **-archive** flag.

If you are using a non-default port, you must specify the **-p** flag with the port number of the JMS IQ Manager.

The following example shows an example of **-backup**:

```
stcmsctrlutil -backup c:\eGate\client\Archiver\Ar09112003.zip "09/11/
2003"
Backup finished. Archived messages: 2003
```

9.3.16 Browsing Archives

When you have backed up the topics and queues for a particular date, you can browse the archive (a .zip file) with the **-archive** flag.

The **-archive** flag functions similarly to **-journaler** except that you specify the directory where you unzipped the archive. The MS Control utility displays information in the same format as **-journaler**.

The following example displays the same information as **-journaler**, but it reads this information from c:\eGate\client\Archiver\backup.

```
stcmsctrlutil -ar c:\eGate\client\Archiver\backup -tl
Number Of Topic(s): 1
Topic List:
        T0
```

The following example displays information about topic T0.

```
stcmsctrlutil -ar c:\eGate\client\Archiver\backup -ts T0
Topic Name: T0
First sequence number: 0
Last sequence number: 1000
First enqueue time: 09122003:00:14:17
Last enqueue time: 09122003:00:14:00
Number of current subscribers: 0
Number of total subscribers: 0
Message count: 1001
Lowest subscriber sequence: 0
Highest subscriber sequence: 0
```

The following example displays a description of the message with sequence number 1 from the archive c:\eGate\client\Archiver\backup.

```
stcmsctrlutil -ar c:\eGate\client\Archiver\backup -tml T0 1 1
Number Of Messages(s): 1
Message[1]:
Message.SeqNo=1
Message.Timestamp=1031789654330
Journaler.ExpirationTime=1031809654330
Message.Size=228
Message.JMSProperty.EX=0
Message.JMSProperty.TS=1031789654330
Message.JMSProperty.DM=1
```

```
Message.JMSProperty.TY=ASCII
Message.JMSProperty.PR=0
Message.JMSProperty.MI=ID:45c:3b742aa6:950:0a01beee:3d7fdc5614a
Message.JMSProperty.RD=false
Message.UserProperty.JMS_ProducerID=BENCH
```

9.3.17 Setting Timeout

You can use the **-timeout** flag to increase the timeout for the MS Control utility for retrieving messages. The default timeout is five seconds. If the message is not received within five seconds, the utility exits and you see the message "Timeout to receive message from the server, exiting stcmsctrlutil API." This may happen when the JMS IQ Manager is busy. Increasing the timeout as shown below may resolve this problem.

The following example illustrates how you increase the timeout to 15 seconds.

```
stcmsctrlutil -j -tl -timeout 15
```

If the **-timeout** flag is not used in subsequent commands, the default timeout is used.

Appendix A

Additional Java CAPS Message Servers

Additional Java CAPS Message Servers provides information regarding JMS message servers other than the JMS IQ Manager that are included in eGate Integrator.

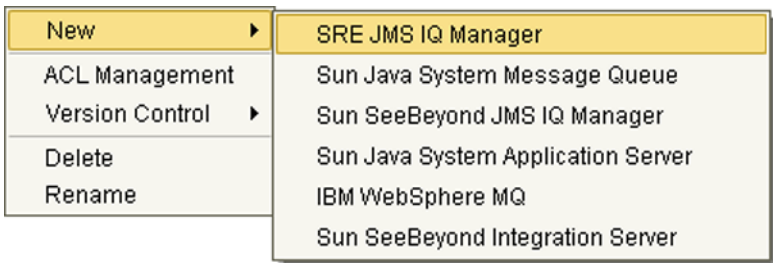
What's in This Appendix

A.1 Using SRE JMS IQ Managers

The SRE JMS IQ Manager allows JMS clients implemented in Java CAPS to connect to JMS servers implemented in release 4.5.x (SRE) of eGate Integrator.

You must install the SRE JMS IQ Manager as an individual eGate component, and then add an instance to the Logical Host in Enterprise Designer (see Figure 26). You enter the connection information (host name and port number) in the SRE JMS IQ Manager properties in the Java CAPS Environment, as described in the *Sun SeeBeyond eGate Integrator User's Guide*. To connect a Java CAPS JMS client to a version 4.5.x JMS server at run time, the SRE JMS Message Server must already be running.

Figure 26 Logical Host Context Menu



The following procedure describes how to install the SRE JMS IQ Manager in a previously-installed eGate Integrator system. For detailed, general installation instructions, refer to the *Sun Java™ Composite Application Platform Suite Installation Guide.*

To install the SRE message server .sar file

1. In the **Administrator** page of the Java CAPS Installer, click **Click to install additional products**.
2. In the **Administrator > Select** page, expand **Core Product**, select **JMSClientToSREJMSIQMgr** and click **Next**.
3. In the **Administrator > Upload** page, select the requested **.sar** files and click **Next**.

   When the installation is finished, the "Installation Completed" message appears.

A.2 Using Sun Java System Message Queues

Java CAPS Projects can be deployed to the Sun Java Enterprise System (JES), by setting up a Logical Host/domain containing a Sun Java System Application Server and a Sun Java System Message Queues. It is important that you be aware of the following issues when working with the Sun Java System Message Queue (SJSMQ).

A.2.1 Compatibility

- The connector has been certified with client runtime version **3.7UR1**. When deploying into Sun Java System Application Server 8.2, make sure that the version of the **imqjmsra.jar** file in the classpath of the server is of this version or higher.
- Both SJSMQ JMS and SSLJMS services are supported.

A.2.2 URL Syntax

- The required URL syntax is:

  ```
  mq://host:port/serviceName?option1=value1&option2=value2.
  ```

- If **serviceName** is omitted, JMS service will be used as default.
- To support one or more Sun Java System Message Queues addresses, add a comma and another URL, for example:

  ```
  mq://host:port/serviceName?option1=value1&option2=value2, mq://
  host:port/serviceName?option1=value1&option2=value2
  ```

- Options can include options for JMSJCA and options for the client runtime. The latter options are propagated to the connection factories.

A.2.3 Concurrency

- The connector supports only sync concurrency mode; concurrent processing is unavailable for topics.
- To enable concurrent processing on queues, make sure to provide the following parameter in the configuration of the Sun Java System Message Queues broker:

```
imq.autocreate.queue.maxNumActiveConsumers=-1
```

- The default number of messages is sent to one queue-receiver; since this is likely to be too many to permit reasonable concurrency, it is recommended to set the following additional parameter in the broker configuration:

```
imq.autocreate.queue.consumerFlowLimit=10
```

A.2.4 Message Management

- The management message-driven bean (MBean) for management of messages in queues and topics is limited to queues only, because the client runtime does not provide functionality for the management of topics.
- To make management messages persistent, add the following parameter in the configuration file of the SJSMQ broker:

```
imq.metrics.topic.persist=true
```

- Refer to the documentation of the Sun Java System Message Queues for a description of this last parameter, as well as others such as:

```
imq.metrics.interval
imq.metrics.topic.interval
imq.metrics.topic.timetolive
```

A.3 Using Sun JMS Grid Clients

This section describes how to:

- Install the JMS Grid client to work with Sun SeeBeyond Enterprise Designer
- Add a JMS Grid client to an eGate Environment
- Configure a JMS Grid client

Note: *This section applies only to a JMS Grid client connection; the server is configured separately, using the JMS Grid administration tool.*

A.3.1 Installing the JMS Grid Client

Installing consists of uploading the JMS Grid archive file to the Repository, and then using the Enterprise Designer Update Center to import the necessary files into Enterprise Designer. The following procedure assumes that eGate Integrator, including Enterprise Designer, has been installed.

Note: *Uploading **.sar** files and using the Enterprise Designer Update Center are described in detail in the Java Composite Application Platform Suite Installation Guide.*

To install the JMS Grid client into Enterprise Designer

1. On the **JMSGrid_CAPS** DVD, select the file **JMS_Grid.sar** and upload it to the Repository.

2 Start Enterprise Designer and select **Update Center** from the **Tools** menu.

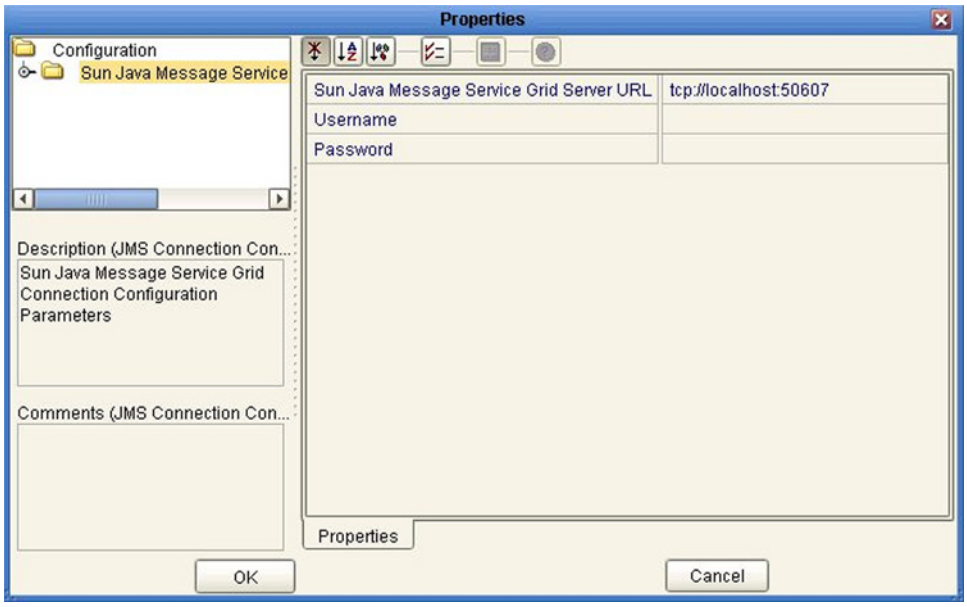
3 Use the **Check for Available Updates** option to locate **Sun Java Message Service Grid plug-in for ED** and install the module as described in the *Sun Java™ Composite Application Platform Suite Installation Guide*.

4 After restarting Enterprise Designer (IDE), the **Sun Java Message Service Grid** will be available as a message server option for the Logical Host.

A.3.2 Configuring the JMS Grid Client

Selecting **Properties** from the JMS Grid context menu in Environment Explorer displays the dialog shown in Figure 27.

Figure 27 Sun JMS Grid Client Configuration Properties



The URL must be of the form:

```
tcp://host:port?option1=value1&option2=value2
```

- All JMS Grid protocols are supported (TCP, SSL, HTTP).
- Multiple host/port combinations can be specified denoting the nodes in a cluster:

  ```
  tcp://host:port,tcp://host:port, ...
  ```

  These host/port combinations are propagated to the **messageChannels** property of the JMS Grid client.
- Options can include those for JMSJCA and for the client runtime.

Appendix B

Troubleshooting

Troubleshooting contains descriptions of suggested solutions for problems you may encounter when using the JMS IQ Manager.

What's in This Appendix

B.1 Timestamp Errors

If the timestamps in the **stcms.log** are not correct, given the system time zone (as specified by the TZ environment variable), shut down the domain and edit the **logicalhost/is/domains/<domain name>/config/domain.xml** file, inserting the following line after the line **<!-- various required jvm-options -->**:

```
<jvm-options>-Dstcms.timezone=<your timezone></jvm-options>
```

Example

```
<!-- various required jvm-options -->
<jvm-options>-Dstcms.timezone=PST8PDT</jvm-options>
```

Glossary

Collaboration
A logical operation performed between some combination of message destinations and external applications. The operation is defined by a Collaboration Definition (see next entry), which can be encoded in either Java or XSLT.

Collaboration Definition
The encoding of business rules, in Java or XSLT format. Typically, the encoding consists of operations on an **Object Type Definition (OTD)**. Several Collaborations can have the same Collaboration Definition.

Connection
Consists of the configuration information that enables an eWay to connect to an external system.

Connectivity Map
Contains business logic and routing information about the data transmission. A Connectivity Map usually includes one or more **Collaboration**s, **Topic**s, **Queue**s, and **eWay**s. A Connectivity Map is created under a **Project**. A Project may have multiple Connectivity Maps.

Deployment Profile
Contains the information about how the **Project** components will be deployed in an Environment. A Project can have multiple Deployment Profiles, but only one Deployment Profile can be activated for a Project in any one **Environment**.

Derived Collaboration
A Collaboration that inherits operations from another, according to standard object-oriented practice.

Domain
A domain contains the eGate Integrator run-time components, including integration servers and message servers, that are installed on a host hardware platform. It is an instance of a **Logical Host**.

Enterprise Designer
The **Project** design tool within eGate Integrator.

Environment
A collection of physical resources and their configurations that are used to host eGate Integrator **Project** components. An Environment contains Logical Hosts and external systems.

eWay
A link between a **Collaboration** and an external connection including the message server connection (topic or queue) or external application.

External Application
A logical representation of an application external to the Java™ Composite Application Platform Suite.

External System
A representation of a computer system hosting an application external to the Java™ Composite Application Platform Suite.

Integration Server
J2EE-compatible software platform that houses the business logic container used to run Collaborations and JCA connectors (eWays). Provides transaction services, persistence, and external connectivity.

JMS IQ Manager
JMS-compliant, guaranteed delivery store, forwarding, and queueing service.

JMSJCA
An implementation of the Java Connector Architecture version 1.5, used to integrate JMS providers with J2EE application servers.

Link
The JMS Connection between a **Collaboration** and a topic or queue in a JMS-compliant message server.

Linked Message Destination
A reference to a **Message Destination** defined in another **Connectivity Map**.

Logical Host
A Logical Host is a template for a **Domain**, which contains the eGate Integrator run-time components.

Message Destination
A general term for a topic or queue. Two or more Projects can share a message destination that has the same name and is deployed on the same message server. A single Project may also have a single message destination referenced in multiple **Connectivity Map**s.

Object Type Definition (OTD)
Object Type Definitions contain the data structure and rules that define an object. OTDs are used in **Collaboration Definition**s for creating data transformations and interfacing with external systems.

Project
Contains a collection of logical components, configurations, and files that are used to solve business problems. A Project organizes the files and packages and maintains the settings that comprise an eGate Integrator system in **Enterprise Designer**.

Queue

A JMS queue is a shareable object that conforms to the *point-to-point* (p2p, or PTP) messaging domain, where one sender delivers a message to exactly one receiver. When the Sun SeeBeyond **JMS IQ Manager** sends a message to a queue, it ensures it is received once and only once, although there may be many receivers "listening" to the queue. This is equivalent to the subscriber pooling in other queue implementations. You can reference a queue that exists in another **Connectivity Map** or **Project**.

Repository

Stores and manages the setup, component, and configuration information for eGate Integrator **Project**s. The Repository also provides monitoring services for Projects, which include version control and impact analysis.

Schema Runtime Environment (SRE)

An add-on feature in eGate Integrator 5.0 that allows **Collaboration**s developed in e*Gate 4.x to be used in and controlled from eGate Integrator 5.0, thereby providing an interim upgrade path for e*Gate 4.x users.

Service

Contains the information about executing a set of business rules. These business rules can be defined in a Java or XSLT **Collaboration Definition**, Business Process, eTL Definition, or other service. A Service also contains binding information for connecting to JMS **Topic**s, **Queue**s, **eWay**s, and other services.

Topic

A JMS topic is a shareable object that conforms to the *publish-and-subscribe* (pub/sub) messaging domain, where one publisher broadcasts messages to one or more subscribers. When the Sun SeeBeyond **JMS IQ Manager** publishes a message on a topic, it ensures that all subscribers receive the message.

XA is a vendor-neutral protocol that was devised to manage transactions between multiple client application programs and multiple database systems. Documentation on distributed transaction processing using XA is available at no charge from The Open Group at **http://www.opengroup.org** (search on "XA").

Index

F

I

J

L

M

X