G2 JMSLink

User's Guide

Version 2.3 Rev. 1

G2 JMSLink User's Guide, Version 2.3 Rev. 1

May 2020

The information in this publication is subject to change without notice and does not represent a commitment by Gensym Corporation.

Although this software has been extensively tested, Gensym cannot guarantee error-free performance in all applications. Accordingly, use of the software is at the customer's sole risk.

Ignite Technologies, Inc.
401 Congress Ave., Suite 2650
Austin, TX 78701 USA
Telephone: +1-800-248-0027
Email: success@ignitetech.com

Part Number: DOC131-230

Contents

Preface

Describes this guide and the conventions that it uses.

About this Guide

This guide explains how to use the G2 JMSLink bridge to connect G2 applications to Java Message Service (JMS). By connecting to JMS, G2 applications can interact with native message-oriented middleware (MOM) systems, which are designed especially for enterprise messaging applications.

Audience

To use this manual, you must have at least a limited knowledge of G2 and a thorough understanding of JMS.

Conventions

This guide uses the following typographic conventions and conventions for defining system procedures.

Typographic

Convention Examples	Description
g2-window, g2-window-1, ws-top-level, sys-mod	User-defined and system-defined G2 class names, instance names, workspace names, and module names
history-keeping-spec, temperature	User-defined and system-defined G2 attribute names
true, 1.234, ok, “Burlington, MA”	G2 attribute values and values specified or viewed through dialogs
Main Menu > Start KB Workspace > New Object create subworkspace Start Procedure	G2 menu choices and button labels
conclude that the x of y ...	Text of G2 procedures, methods, functions, formulas, and expressions
new-argument	User-specified values in syntax descriptions
text-string	Return values of G2 procedures and methods in syntax descriptions
File Name, OK, Apply, Cancel, General, Edit Scroll Area	GUIDE and native dialog fields, button labels, tabs, and titles
File > Save Properties	GMS and native menu choices
workspace	Glossary terms

Convention Examples	Description
`c:\Program Files\Gensym\`	Windows pathnames
`/usr/gensym/g2/kbs`	UNIX pathnames
`spreadsh.kb`	File names
`g2 -kb top.kb`	Operating system commands
`public void main()` `gsi_start`	Java, C and all other external code

Note Syntax conventions are fully described in the *G2 Reference Manual*.

Procedure Signatures

A procedure signature is a complete syntactic summary of a procedure or method. A procedure signature shows values supplied by the user in *italics*, and the value (if any) returned by the procedure *underlined*. Each value is followed by its type:

g2-clone-and-transfer-objects
(*list*: class item-list, *to-workspace*: class kb-workspace,
delta-x: integer, *delta-y*: integer)
-> *transferred-items*: g2-list

Related Documentation

G2 Core Technology

- *G2 Bundle Release Notes*
- *Getting Started with G2 Tutorials*
- *G2 Reference Manual*
- *G2 Language Reference Card*
- *G2 Developer? Guide*
- *G2 System Procedures Reference Manual*

- *G2 System Procedures Reference Card*
- *G2 Class Reference Manual*
- *Telewindows User? Guide*
- *G2 Gateway Bridge Developer? Guide*

G2 Utilities

- *G2 ProTools User? Guide*
- *G2 Foundation Resources User? Guide*
- *G2 Menu System User? Guide*
- *G2 XL Spreadsheet User? Guide*
- *G2 Dynamic Displays User? Guide*
- *G2 Developer? Interface User? Guide*
- *G2 OnLine Documentation Developer? Guide*
- *G2 OnLine Documentation User? Guide*
- *G2 GUIDE User? Guide*
- *G2 GUIDE/UIL Procedures Reference Manual*

G2 Developers' Utilities

- *Business Process Management System User? Guide*
- *Business Rules Management System User? Guide*
- *G2 Reporting Engine User? Guide*
- *G2 Web User? Guide*
- *G2 Event and Data Processing User? Guide*
- *G2 Run-Time Library User? Guide*
- *G2 Event Manager User? Guide*
- *G2 Dialog Utility User? Guide*
- *G2 Data Source Manager User? Guide*
- *G2 Data Point Manager User? Guide*
- *G2 Engineering Unit Conversion User? Guide*
- *G2 Error Handling Foundation User? Guide*
- *G2 Relation Browser User? Guide*

Bridges and External Systems

- *G2 ActiveXLink User? Guide*
- *G2 CORBALink User? Guide*
- *G2 Database Bridge User? Guide*
- *G2-ODBC Bridge Release Notes*
- *G2-Oracle Bridge Release Notes*
- *G2-Sybase Bridge Release Notes*
- *G2 JMail Bridge User? Guide*
- *G2 Java Socket Manager User? Guide*
- *G2 JMSLink User? Guide*
- *G2 OPCLink User? Guide*
- *G2 PI Bridge User? Guide*
- *G2-SNMP Bridge User? Guide*
- *G2 CORBALink User? Guide*
- *G2 WebLink User? Guide*

G2 JavaLink

- *G2 JavaLink User? Guide*
- *G2 DownloadInterfaces User? Guide*
- *G2 Bean Builder User? Guide*

G2 Diagnostic Assistant

- *GDA User? Guide*
- *GDA Reference Manual*
- *GDA API Reference*

Customer Support Services

You can obtain help with this or any Gensym product from Gensym Customer Support. Help is available online, by telephone and by email.

To obtain customer support online:

➔ Access Ignite Support Portal at `https://support.ignitetech.com`.

You will be asked to log in to an existing account or create a new account if necessary. Ignite Support Portal allows you to:

- Register your question with Customer Support by creating an Issue.
- Query, link to, and review existing issues.
- Share issues with other users in your group.
- Query for Bugs, Suggestions, and Resolutions.

To obtain customer support by telephone or email:

➔ Use the following numbers and addresses:

United States Toll-Free	+1-855-453-8174
United States Toll	+1-512-861-2859
Email	support@ignitetech.com

1

Introduction

Provides a general description of JMS and G2 JMSLink.

Introduction

JMS, an acronym for Java Message Service, is an industry-standard API for Java-based clients to interact with native message-oriented middleware (MOM) systems, which are designed especially for enterprise messaging applications.

Some JMS-compliant MOM systems include IBM's WebSphereMQ, Sun JMQ, FioranoMQ, BEA Weblogic, and JBoss 3.2.6 on Linux and HP, Open JMS 0.7.6.1 on Windows, Linux, and HP, Java 2 Platform, Enterprise Edition (J2EE) 1.3.1 on Linux.

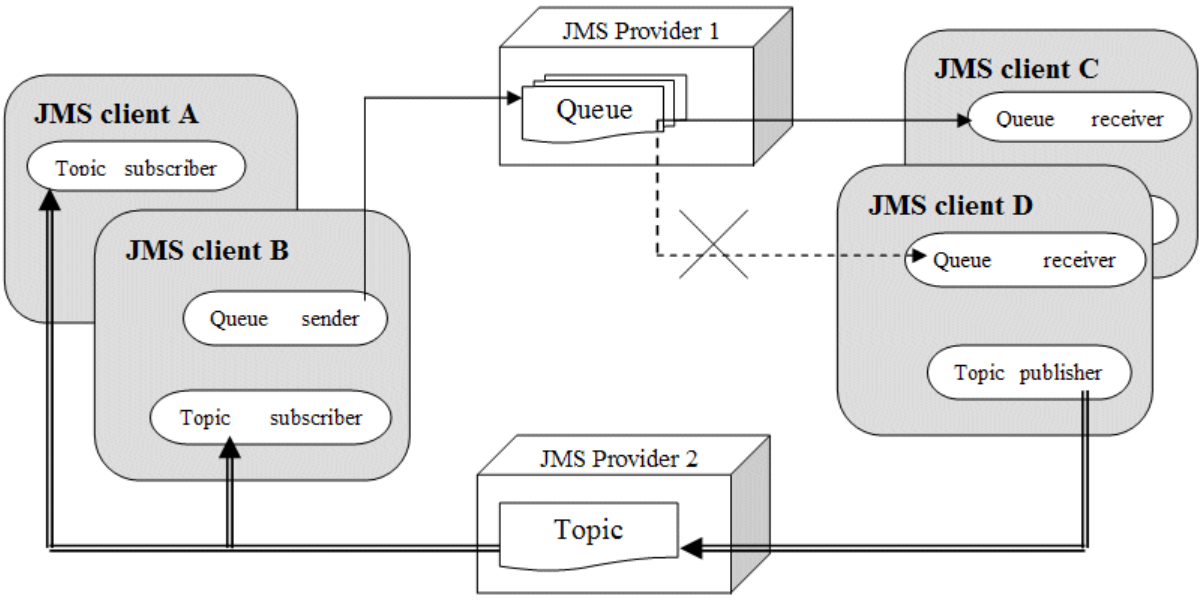
A key concept of enterprise messaging is that messages are delivered asynchronously from one system to another over a network. A major goal of JMS is for clients to have a consistent API for creating and working with messages, which is independent of multiple vendor-specific JMS providers.

JMS supports two types of messaging models:

- Point-to-point (PTP) is a one-to-one message delivery system that allows only two JMS clients to send and receive messages, both synchronously and asynchronously, via a virtual channel known as a queue.
- Publish-and-subscribe (pub/sub) is a one-to-many message delivery system in which one JMS client is a message publisher that can send a message to many JMS clients as message subscribers through a virtual channel known as a topic.

JMS providers ensure guaranteed messaging, meaning that intended message consumers eventually receive a message, even if partial failure occurs. A partial failure can occur, for example, when one of the networked systems might have an unpredictable failure or need to be shut down at some time during its continuous operation. The guaranteed messaging offers guaranteed message delivery, message acknowledgement, message group acknowledgment, and transacted messages.

The following figure shows an example of a B2B application in which JMS clients communicate with each other via two individual JMS-compliant messaging systems: JMS provider 1 and JMS provider 2. JMS client B sends a message to a queue in JMS provider 1, but only one of the potential receivers, client C, receives the message. JMS client D publishes a message to a topic in JMS provider 2, and all subscribers to the topic, JMS client A and JMS client B, receive the message.



Common JMS Terms

Here are some commonly used JMS terms that you should be familiar with:

Term	Definition
JMS provider	A messaging product that implements the JMS specification.
JMS message	A package of business data that carries all of the data and state needed by the business logic that processes it. As of JMS Version 1.0.2b, six message interface types must be supported by JMS providers. The six message interfaces are: `Message`, and its five sub-interfaces, `TextMessage`, `StreamMessage`, `MapMessage`, `ObjectMessage`, and `BytesMessage`.
JMS client	A computer program that can produce and/or consume JMS messages to/from a JMS provider.
JMS destination	A message destination object, either a topic or queue, created within a JMS provider, which are the places where JMS clients can send and receive JMS messages to and from the JMS provider.
JMS application	A set of applications that define JMS messages and a set of JMS clients that exchange those messages.

Features of G2 JMSLink

G2 JMSLink provides the following features.

Data Mapping between G2 and JMS Providers

The data mapping for primitive data values between G2 and JMS providers is based on the data mapping between G2 and G2 JavaLink. For details, see the *G2 JavaLink User? Guide*.

Message Types

G2 JMSLink supports two JMS message types: `TextMessage` and `MapMessage`. For an instance of either `TextMessage` or a subclass of `TextMessage`, the message body is mapped to a G2 text value. For an instance of either `MapMessage` or a subclass of `MapMessage`, the message body is mapped to a G2 structure value.

G2 JMSLink listens to either a topic or a queue of a JMS provider for any new `TextMessage` or `MapMessage` messages. Upon receiving an instance of either class, the bridge converts the data according to its type and passes the message to the G2 server. For the bridge to process a `MapMessage`, each name-value pair is converted to a name-value pair in a G2 structure. For primitive data values, the data conversion is based on the G2 JavaLink data mapping.

Message Headers

Every JMS message has a set of standard headers. The current version of G2 JMSLink supports most of the standard headers except `JMSCorrelationIDAsBytes` as a byte array, and `JMSDestination` and `JMSReply`, both of which identify the destination with an administered object, either a `Topic` or `Queue`.

G2 JMSLink supports these message header types:

- `JMSDeliveryMode`
- `JMSMessageID`
- `JMSTimestamp`
- `JMSExpiration`
- `JMSRedelivered`
- `JMSPriority`
- `JMSCorrelationID`
- `JMSType`

Message Properties

Properties act like additional headers that can be assigned to a message. They provide the developer with more information about the message.

The three basic categories of message properties are: application-specific properties, JMS-defined properties, and provider-specific properties.

The current version of G2 JMSLink supports application-specific properties only. Property values can be any G2 value type: integer, float, text, symbol, and truth-value.

JMS-defined properties have the same characteristics as application-specific properties, except that most of them are set by the JMS provider when the message is sent. Supported JMS-defined properties vary from vendor to vendor.

For further details on message properties, see Appendix A, Message Properties.

Message Selectors

G2 JMSLink supports message selectors that allow a JMS consumer to be more selective about the messages it receives from a particular destination, either a `Topic` or `Queue`. Message selectors use message properties and headers as criteria in conditional expressions. The conditional expressions use boolean logic to declare which message should be delivered to a JMS consumer.

For further details on message selectors, see Appendix B, Message Selectors.

Durable Subscriptions

A durable subscription is one that outlasts a client's connection with a message server. When a durable subscriber is disconnected from the JMS server, it is the responsibility of the server to store messages that the subscriber misses. When the durable subscriber reconnects, the message server sends all the unexpired messages that have accumulated.

Asynchronous Messaging Model

G2 JMSLink supports two asynchronous messaging models: Point-to-Point (p2p) and Publish-and-Subscribe (pub/sub). For more information, see the Introduction.

Guaranteed Messaging

The three main features of guaranteed messaging are message autonomy, store-and-forward, and the underlying message acknowledgment semantics.

Required Software

G2

The G2 JMSLink knowledge base, `jms.kb`, is compatible with G2 Version 7.0 Rev. 1 or later on any platform where G2 is supported.

G2 JavaLink

G2 JMSLink is compatible with G2 JavaLink Version 1.2 Rev. 6 or later. Refer to the G2 JavaLink readme file for software and system requirements.

Java Message Service (JMS)

G2 JMSLink is compatible with any JMS providers that implement JMS specification Version 1.0.2b.

This release has been tested with IBM's WebSphereMQ Version 5.3.1, JBoss Version 3.2.1, J2EE 1.3.1 Reference Implementation, FioranoMQ Version 7.0, and OpenJMS 0.7.6.1.

If you are using a different JMS provider, you need to create a custom batch file to start the bridge. For more information, see Running G2 JMSLink.

On WebSphere MQ, the function of this SupportPac (MA0C) is incorporated in WebSphere MQ V5.3 by Fix Pack 8 for the following platforms: Windows, AIX, Solaris, HP-UX, Linux for Intel. Support for the `pubsub` function is now supplied as a Fix Pack on each of these platforms. The Fix Packs can be found by following the links from `http://www.ibm.com/software/integration/mqfamily/support/summary/`

Do not install this SupportPac on systems that already have Fix Pack 8 or later applied. Also, note that the Fix Packs are cumulative, so you can just install the latest Fix Pack, which is Version 9 for MQ V5.3.

Java/JDK

G2 JMSLink requires J2SE Version 1.4 or later.

2

Getting Started

Describes the basics of how to interact with G2 JMSLink.

Introduction

This chapter describes how to install, run, configure, connect, and disconnect G2 JMSLink.

Installing G2 JMSLink and JMS

You install G2 JMSLink as part of the G2 Bundle or as part of any Gensym application bundle that contains the G2 Bundle.

To run G2 JMSLink, you must have a vendor-specific JMS provider properly installed and running first. G2 JMSLink has been testing with and provides menu choices in the Start for the following JMS vendors:

- JMS FioranoMQ
- JMS J2EE
- JMS JBoss
- JMS WebSphereMQ
- OpenJMS

Note If you encounter problems with JMS WebsphereMQ while running the JMS bridge, refer to HelpLink Resolution HQ-4642733 for information on how to configure WebSphereMQ to run with the G2 JMSLink bridge.

Running G2 JMSLink

The default TCP port number for G2 JMSLink is 22070. You can use the default port or specify your own port by providing an argument to the appropriate batch file. For example, to communicate with G2 JMSLink and IBM WebSphereMQ, you can connect to port 22077.

The batch files that start the bridge are JMS provider-specific, which means you might need to edit the file to change the settings for environment variables to use the correct installation path. See your system administrator for details about the installation directory of your JMS provider.

To start G2 JMSLink on the default port:

➔ Choose Start > Programs > G2 8.3r0 > Bridges > JMS and choose the version of JMS that you are running.

The shortcut is similar for other application bundles except that G2 8.3r0 is replaced by the name of your installed bundle.

To start G2 JMSLink on a different port:

➔ Connect to a command prompt and run the appropriate batch file for your JMS vendor and provide the port number as an argument.

For example, to start G2 JMSLink on a different TCP port from IBM's WebSphere MQ, run this command in a command window:

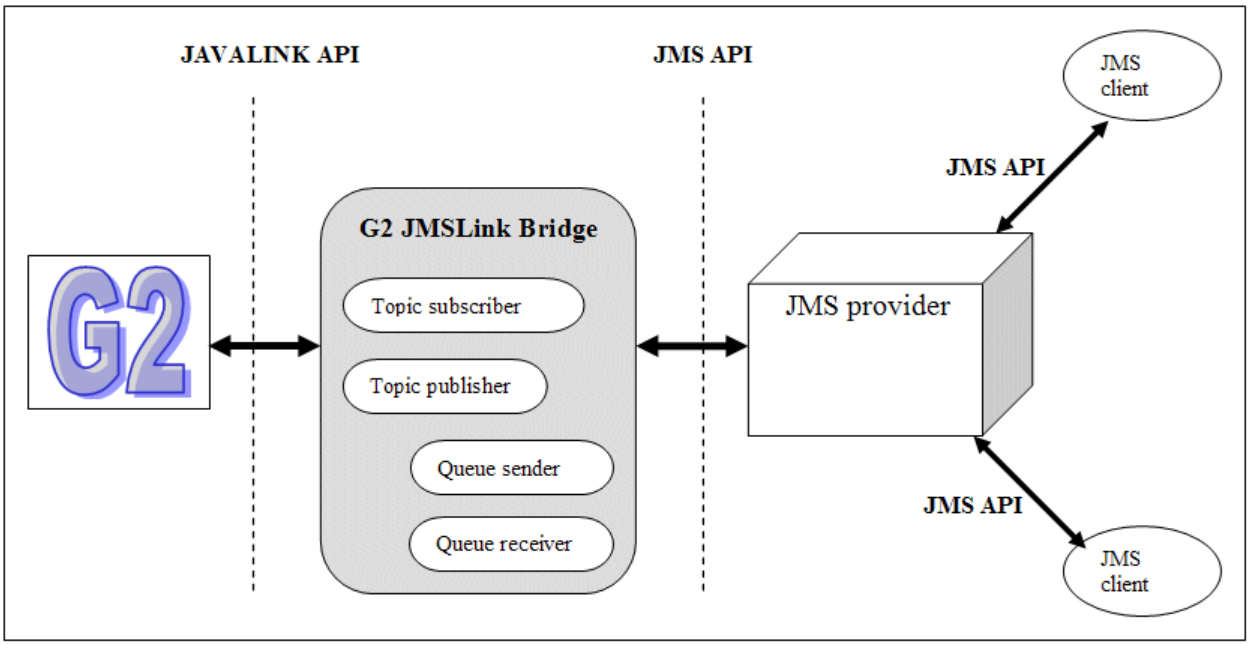
```
\jms\bin\StartJmsBridge-WebSphereMQ.bat 22077
```

Using G2 JMSLink

To use the functionality provided by the G2 JMSLink Bridge in your G2 application, you need to:

- Create a jms-interface object.
- Configure the jms-interface attributes. For a complete description of these attributes, see JMS Interface Attributes.
- Connect to the bridge. For details, see Connecting G2 to the Bridge Process.
- Call the appropriate APIs for point-to-point or publish/subscribe messaging. For a description of the available APIs, see API Reference.

The following figure illustrates how a G2 application exchanges JMS messages with multiple JMS clients by sending and receiving JMS messages to and from a JMS provider via the G2 JMSLink Bridge.



G2 JMSLink includes the `jms-demo.kb` located in the `kbs` directory. This KB provides a number of sample configurations for various vendor-specific JMS providers. Refer to this KB for information on configuring G2 JMSLink.

We recommend that you merge `jms-demo.kb` into your G2 application and test the bridge to communicate with your vendor-specific JMS provider.

To access the example:

➔ Choose Examples > JMSLink Tutorial from the Start menu of your bundle installation.

Connecting G2 to the Bridge Process

You connect your G2 application to the G2 JMSLink bridge process through an instance of jms-interface, either interactively, using a menu choice, or programmatically, using an API procedure.

For details on the API procedure, see Connectivity.

To connect G2 to G2 JMSLink:

➔ Choose Connect to JMS from the popup menu for the instance.

or

➔ Call jms-connect on an instance of jms-interface.

Shutting Down G2 JMSLink

You disconnect the current connection to a bridge and terminate the bridge process through an instance of jms-interface, either interactively, using a menu choice, or programmatically, using an API procedure.

Do not shut down the bridge by closing the command window or entering Ctrl + C in the command window; otherwise, JMS-related resources are not properly recycled.

For details on the API procedure, see Connectivity.

To shut down G2 JMSLink:

➔ Choose Kill Bridge to JMS from the popup menu for the instance.

or

➔ Call jms-kill-bridge on an instance of jms-interface.

3

API Reference

Describes the API procedures for communicating with JMS.

Introduction

This chapter describes the G2 JMSLink Application Programmer's Interface (API). The APIs are divided into these functional categories:

- Connectivity
- Point-To-Point (PTP) Messaging
- Publish / Subscribe (Pub/Sub) Messaging
- Message Handling
- Error Handling

The `jms.kb` contains the jms-interface class definition and its associated API in the form of G2 methods and procedures for communicating with JMS provides.

JMS Interface Attributes

You configure the following attributes of a jms-interface object:

Attribute	Description
names	A unique name for the jms-interface object.
Allowable values:	Any symbol
Default values:	none
Notes:	You do not have to configure this attribute.
gsi-connection-configuration	The network protocol that G2 uses to communicate with the bridge.
Allowable values:	tcp-ip host "*hostname*" port-number *tcp-ip-port-number*
Default values:	none
Notes:	TCP/IP is the only protocol that the bridge supports.
remote-process-initialization-string	A parameter to disable the reporting of messages on the console window where you started the bridge process.
Allowable values:	"-debug" ""
Default value:	"" (empty string): Which disables the display of information messages on the console.
gsi-interface-status	A value that indicates the current status of the connection between the bridge and the G2 application.

Attribute	Description
Allowable values:	2 (OK): The connection has been made and is active. 1 (Initializing): The external system is being initialized. When G2 receives this code, it refrains from sending messages to the bridge until it receives the OK code (2). 0 (Inactive): The connection is disabled or inactive. -2 (Error): An error has occurred, and the connection is broken.
Default value:	0
Notes:	For more information, see the *G2 Gateway User? Guide*. The gsi-interface-status is different from the jms-provider-connection-status described below.
jms-provider	The name of a vendor-specific JMS provider to which the bridge connects.
Allowable values:	Any text
Default values:	"" (empty string)
Notes:	This attribute read only.
jms-initial-context-factory	The value of the `initial_context_factory` used by the naming service for creating an `InitialContext` object.
Allowable values:	Any text
Default values:	"unspecified"
Notes:	This value differs for each provider. Ask your system administrator or check the relevant product documentation for your JMS provider.
jms-provider-url	The value of the `provider_url` used by the naming service for creating an `InitialContext` object.
Allowable values:	Any text
Default values:	"unspecified"

Attribute	Description
Notes:	This value differs for each provider. Ask your system administrator or check the relevant product documentation for your JMS provider.
jms-topic-connection-factory	The value used by the naming service to look up for an instance of `TopicConnectionFactory`.
Allowable values:	Any text
Default values:	"unspecified"
Notes:	You must create and configure a `TopicConnectionFactory` for your vendor-specific JMS provider for the bridge to connect correctly.
jms-queue-connection-factory	The value used by the naming service to look up for an instance of `QueueConnectionFactory`.
Allowable values:	Any text
Default values:	"unspecified"
Notes:	You must create and configure a `QueueConnectionFactory` for your vendor-specific JMS provider for the bridge to connect correctly.
jms-destination-type	A symbol that identifies the type of a JMS destination object to which the bridge registers as a listener.
Allowable values:	topic queue
Default values:	topic
Notes:	Values other than topic or queue are invalid.

Attribute	Description
jms-input-destination-name	The name of the naming service to look up for the JMS destination object (topic or queue) to which the bridge registers for incoming messages.
Allowable values:	Any text
Default values:	"" (empty string)
Notes:	You must create and configure the JMS destination object of a vendor-specific JMS provider for the bridge to connect correctly.
jms-input-destination-selector	A boolean logic that specifies which messages should be delivered to the bridge by the JMS provider.
Allowable values:	Any text
Default values:	""
Notes:	See Appendix B, Message Selectors.
jms-durable-topic-subscription	Whether the subscription to a topic is durable or not.
Allowable values:	true false
Default values:	false
jms-durable-subscription-name	A unique name for the durable subscription.
Allowable values:	Any text
Default values:	"unspecified"
Notes:	The name is provider-specific.

Attribute	Description
jms-input-messages	A temporary buffer for storing all received JMS messages.
Allowable values:	sequence
Default values:	sequence()
Notes:	It is your responsibility to clear messages from the temporary buffer periodically to prevent potential memory leaks. See jms-remove-all-messages.
jms-input-message-procedure-callback	The name of a procedure to use as a message handler.
Allowable values:	Any symbol
Default values:	jms-default-message-handler
Notes:	Specify your customized message handler here.
jms-bridge-error-message-procedure-callback	The name of a procedure to use as a error handler.
Allowable values:	Any symbol
Default values:	jms-default-bridge-error-handler
Notes:	Specify your customized error handler here.
jms-output-destination-name	The name of the naming service to look up for the JMS destination object (topic or queue) to which the bridge registers for outgoing messages.
Allowable values:	Any text
Default values:	"" (empty string)
Notes:	You must create and configure the JMS destination object of a vendor-specific JMS provider for the bridge to connect correctly.

Attribute	Description
jms-topic-receive-local-copy	Whether the bridge receives a copy of the message that it sends to a topic.
Allowable values:	true false
Default values:	false
jms-acknowledge-mode	The acknowledgement mode to use, which can be one of these symbols: • auto_acknowledge — Automatically acknowledges the receipt of a message. • client_acknowledge — Acknowledges the receipt of a message conditionally by calling acknowledge method. • dups_ok_acknowledge — Allows the delivery of duplicate messages. To use the mode client_acknowledge: • For a "queue" interface object, before connecting to a JMS bridge, configure the jms-acknowledge-mode to be the symbol client_acknowledge. • For a "topic" interface object, before connecting to a JMS bridge, configure the jms-acknowledge-mode to be the symbol client_acknowledge and configure the jms-durable-topic-subscription to be true. • Customize the end user's message handler to handle each incoming message. To acknowledge the receipt of an incoming message, the handler should return true; otherwise, the handler should return false.
Allowable values:	symbol
Default values:	auto_acknowledge
Notes:	For a detailed description of these options, see the JMS specification 1.0.2b.

Attribute	Description
jms-transacted-delivery	Whether the message is transactional.
Allowable values:	true false
Default values:	false
Notes:	The current version of the bridge does not support transactional messaging such as commit and rollback.
jms-persistent-delivery	Whether the delivered message is persistent.
Allowable values:	true false
Default values:	true
Notes:	The message persistence is typically overridden by the message header.
jms-message-priority	The message priority that the JMS provider uses to prioritize the delivery of messages.
Allowable values:	0 – 4: gradations of normal priority 5 – 9: gradations of expedited priority
Default values:	4
Notes:	The message priority is typically overridden by the message header.
jms-message-alive-time	The expiration time of a message.
Allowable values:	Any integer
Default values:	0
Notes:	The expiration time is set in milliseconds and is typically overridden by the message header.

Attribute	Description
jms-username	A user name for logging into the JMS provider.
Allowable values:	Any text
Default values:	"" (empty string)
Notes:	This attribute is optional and depends on how your vendor-specific JMS provider is configured.
jms-password	A password for logging into the JMS provider.
Allowable values:	Any text
Default values:	"" (empty string)
Notes:	This attribute is optional and depends on how your vendor-specific JMS provider is configured.
jms-client-id	The unique identification number for durable topic subscription.
Allowable values:	Any text
Default values:	"" (empty string)
Notes:	This attribute is optional and depends on whether the subscription to a topic is a durable subscription.
jms-provider-connection-status	The connection status between the bridge and the JMS provider.
Allowable values:	connected disconnected
Default values:	disconnected
Notes:	See also gsi-connection-status.

Connectivity

This section describes the API methods and procedures used to connect to and disconnect from a JMS provider from a G2 application:

jms-connect

Establishes a connection between G2 and the JMS provider via the G2 JMSLink Bridge.

Synopsis

jms-connect
(*io*: class jms-interface, *win*: class g2-window)
-> (*status*: symbol, *code*: integer, *message*: text)

Argument	Description
io	A gsi-interface to be connected to a G2 JMSLink Bridge process.
win	The G2 window used to display messages.

Return Value	Description
status	One of the following symbols: success, error, warning, info, disconnected, fatal
code	One of the following codes: • 1 — operation successful • 0 — operation failure
message	The operation message.

Description

This method of jms-interface initiates a connection to a bridge listening to a specified TCP port. Once the connection from G2 to the bridge has been established, the JMS interface configuration is passed to the bridge that initiates the connection to the specified JMS provider.

For a list of supported JMS providers, see Java Message Service (JMS).

You can call jms-connect to establish a new connection between G2 and the bridge and a connection between the bridge and a JMS provider. You can also call it to reestablish a connection that has been broken or disconnected between the bridge and the JMS provider.

If a connection already exists between the bridge and the JMS provider when you call jms-connect, the method returns this error: "Operation failed. JMS Interface has been already connected to JMS provider".

jms-disconnect

Disconnects the G2 JMSLink Bridge from the JMS provider.

Synopsis

jms-disconnect
(*io*: jms-interface, *win*: ui-client-item)

Argument	Description
io	The jms-interface object used for connecting to the JMS provider.
win	The G2 window used to display messages.

Description

This method of jms-interface removes the connection between the bridge and the JMS provider that was established by a call to jms-connect. It does not break connections configured by other jms-interface objects, nor does it disconnect the bridge from G2.

This method also cleans up and frees all resources associated with the bridge, such as the JMS connection and session, queue receiver, and topic subscriber.

You can disconnect the bridge from the JMS provider at any time. However, until the connection to the JMS provider is reestablished, G2 cannot produce or consume any JMS messages to or from the JMS provider over that jms-interface object.

jms-kill-bridge

Terminates the G2 JMSLink Bridge that is connected to G2.

Synopsis

jms-kill-bridge
(*io*: jms-interface, *win*: ui-client-item)

Argument	Description
io	The jms-interface object used for connecting to the JMS provider.
win	The G2 window used to display messages.

Description

This method of jms-interface terminates the connected bridge process. Thus, it terminates the connection from G2 to the bridge.

This method also cleans up and frees all resources associated with the bridge, such as the JMS connection and session, queue receiver, and topic subscriber.

You can terminate the bridge at any time. However, until a connection to the JMS provider is reestablished, G2 cannot produce or consume any JMS messages to or from the JMS provider.

Point-To-Point (PTP) Messaging

This section describes the API methods and procedures used for point-to-point messaging. A G2 application can programmatically send a `TextMessage`, a `MapMessage`, or a serialized G2 object to a JMS provider with the following APIs:

jms-send-text-message
jms-send-map-message
jms-send-map-message

jms-send-text-message

Sends a G2 text value to a specified queue of a JMS provider in a `TextMessage` format.

Synopsis

jms-send-text-message
(*io*: jms-interface, *msg*: text , *header*: structure,
prop: structure, *win*: ui-client-item)
-> (*status*: symbol, *code*: integer, *message*: text)

Argument	Description
io	The jms-interface object used for connecting to the JMS provider.
msg	A G2 text value for the message payload. Specify "" for no message.
header	A G2 structure containing name-value pairs for the message headers. Specify structure() for no header.
prop	A G2 structure containing name-value pairs for the message properties. Specify structure() for no properties.
win	The G2 window used to display messages.

Return Value	Description
status	One of the following symbols: success, error, warning, info, disconnected, fatal
code	One of the following codes: • 1 — operation successful • 0 — operation failure
message	The operation message.

Description

This method of jms-interface sends a G2 text value in the format of a JMS `TextMessage` to a queue, a destination object of the JMS provider. The JMS provider is responsible for delivering the JMS `TextMessage` asynchronously to a client that is a message consumer of the queue.

You can call this method at any time provided an established connection exists between the jms-interface object and the JMS provider. If no connection exists between G2 and the JMS provider, the method returns this error: "Cannot send JMS message because JMS provider is not connected".

jms-send-map-message

Sends a G2 structure to a specified queue of a JMS provider in a `MapMessage` format.

Synopsis

jms-send-map-message
(*io*: jms-interface, *msg*: structure, *header*: structure,
prop: structure, *win*: ui-client-item)
-> (*status*: symbol, *code*: integer, *message*: text)

Argument	Description
io	The jms-interface object used for connecting to the JMS provider.
msg	A G2 structure containing name-value pairs for the message payload. Specify structure() for an empty message.
header	A G2 structure containing name-value pairs for the message headers. Specify structure() for no header.
prop	A G2 structure containing name-value pairs for the message properties. Specify structure() for no properties.
win	The G2 window used to display messages.

Return Value	Description
status	One of the following symbols: success, error, warning, info, disconnected, fatal
code	One of the following codes: • 1 — operation successful • 0 — operation failure
message	The operation message.

Description

This method of jms-interface sends a G2 structure in the format of a JMS `MapMessage` to a queue, a destination object of the JMS provider. The JMS provider is responsible for delivering the JMS `MapMessage` asynchronously to a client that is a message consumer of the queue.

You can call this method at any time provided an established connection exists between the G2 jms-interface object and the JMS provider. If no connection exists between G2 and the JMS provider, the method returns this error: "Cannot send JMS message because JMS provider is not connected".

jms-send-map-message

Sends an instance of `MapMessage` represented by two G2 sequences to a specified queue of a JMS provider.

Synopsis

jms-send-map-message
(*io*: jms-interface, *keys*: sequence, *keyvalues*: sequence,
header: structure, *prop*: structure, *win*: ui-client-item)
-> (*status*: symbol, *code*: integer, *message*: text)

Argument	Description
io	The jms-interface object used for connecting to the JMS provider.
keys	A G2 sequence representing names for the `MapMessage` payload. Specify "" for no keys.
keyvalues	A G2 sequence representing values for the names. Specify "" for no keys.
header	A G2 structure containing name-value pairs for the message headers. Specify structure() for no header.
prop	A G2 structure containing name-value pairs for the message properties. Specify structure() for no properties.
win	The G2 window used to display messages.

Return Value	Description
status	One of the following symbols: success, error, warning, info, disconnected, fatal
code	One of the following codes: • 1 — operation successful • 0 — operation failure
message	The operation message.

Description

This method of jms-interface sends a JMS `MapMessage` to a queue, a destination object of the JMS provider. The JMS `MapMessage` containing a set of name-value pairs is represented as two G2 sequences, one containing a list of names and the other containing a list of values. The JMS provider is responsible for delivering the JMS `MapMessage` asynchronously to a client that is a message consumer of the queue.

You can call this method at any time provided an established connection exists between the G2 jms-interface object and JMS provider. If no connection exists between G2 and the JMS provider, the method returns this error: "Cannot send JMS message because JMS provider is not connected".

Publish/Subscribe (Pub/Sub) Messaging

This section describes the API methods and procedures used for publish-and-subscribe messaging. A G2 application can publish a `TextMessage`, `MapMessage` or a serialized G2 object to a JMS provider with the following APIs:

jms-publish-text-message
jms-publish-map-message
jms-publish-map-message

jms-publish-text-message

Publishes a G2 text value to a specified topic of a JMS provider in a `TextMessage` format.

Synopsis

jms-publish-text-message
(*io*: jms-interface, *msg*: text , *header*: structure,
prop: structure, *win*: ui-client-item)
-> (*status*: symbol, *code*: integer, *message*: text)

Argument	Description
io	The jms-interface object used for connecting to the JMS provider.
msg	A G2 text value for the message payload. Specify "" for no message.
header	A G2 structure containing name-value pairs for the message headers. Specify structure() for no header.
prop	A G2 structure containing name-value pairs for the message properties. Specify structure() for no properties.
win	The G2 window used to display messages.

Return Value	Description
status	One of the following symbols: success, error, warning, info, disconnected, fatal
code	One of the following codes: • 1 — operation successful • 0 — operation failure
message	The operation message.

Description

This method of jms-interface publishes a G2 text value in the format of JMS `TextMessage` to a topic, a destination object of the JMS provider. The JMS provider is responsible for delivering the JMS `TextMessage` asynchronously to every client that is a subscriber to the topic.

You can call this method at any time provided an established connection exists between the jms-interface object and the JMS provider. If no connection exists between G2 and the JMS provider, the method returns this error: "Cannot send JMS message because JMS provider is not connected".

jms-publish-map-message

Publishes a G2 structure to a specified topic of a JMS provider in a `MapMessage` format.

Synopsis

jms-publish-map-message
(*io*: jms-interface, *msg*: structure, *header*: structure, *win*: ui-client-item)
-> (*status*: symbol, *code*: integer, *message*: text)

Argument	Description
io	The jms-interface object used for connecting to the JMS provider.
msg	A G2 structure containing name-value pairs for the message payload. Specify structure() for an empty message.
header	A G2 structure containing name-value pairs for the message headers. Specify structure() for no header.
prop	A G2 structure containing name-value pairs for the message properties. Specify structure() for no properties.
win	The G2 window used to display messages.

Return Value	Description
status	One of the following symbols: success, error, warning, info, disconnected, fatal
code	One of the following codes: • 1 — operation successful • 0 — operation failure
message	The operation message.

Description

This method of jms-interface publishes a G2 structure in the format of JMS `MapMessage` to a topic, a destination object of the JMS provider. The JMS provider is responsible for delivering the JMS `MapMessage` asynchronously to every client that is a subscriber to the topic.

You can call this method at any time provided an established connection exists between the G2 jms-interface object and the JMS provider. If no connection exists between G2 and the JMS provider, the method returns this error: "Cannot send JMS message because JMS provider is not connected".

jms-publish-map-message

Publishes an instance of `MapMessage` represented by two G2 sequences to a specified topic of a JMS provider.

Synopsis

jms-publish-map-message
(*io*: jms-interface, *keys*: sequence, *keyvalues*: sequence, *header*: structure, *prop*: sequence, *win*: ui-client-item)
-> (*status*: symbol, *code*: integer, *message*: text)

Argument	Description
io	The jms-interface object used for connecting to the JMS provider.
keys	A G2 sequence representing names for the `MapMessage` payload. Specify "" for no keys.
keyvalues	A G2 sequence representing values for the names. Specify "" for no keys.
header	A G2 structure containing name-value pairs for the message headers. Specify structure() for no header.
prop	A G2 structure containing name-value pairs for the message properties. Specify structure() for no properties.
win	The G2 window used to display messages.

Return Value	Description
status	One of the following symbols: success, error, warning, info, disconnected, fatal
code	One of the following codes: • 1 — operation successful • 0 — operation failure
message	The operation message.

Description

This method of jms-interface publishes a JMS `MapMessage` to a topic, a destination object of JMS provider. The JMS `MapMessage` containing a set of name-value pairs is represented as two G2 sequences, one containing a list of names and the other containing a list of values. The JMS provider is responsible for delivering the JMS `MapMessage` asynchronously to every client that is a subscriber to the topic.

You can call this method at any time provided an established connection exists between the G2 jms-interface object and the JMS provider. If no connection exists between G2 and the JMS provider, the method returns this error: "Cannot send JMS message because JMS provider is not connected".

Message Handling

This section describes the API procedures for managing incoming messages of a jms-interface:

jms-default-message-handler
jms-remove-all-messages

These procedures can be customized specifically for a G2 application.

jms-default-message-handler

Handles incoming messages.

Synopsis

jms-default-message-handler
(*io*: jms-interface, *msg*: value, *header*: structure, *property*: structure, *win*: ui-client-item)
-> *acknowledge-message*: truth-value

Argument	Description
io	The jms-interface object used for connecting to the JMS provider.
msg	A G2 value representing the received JMS message.
header	A G2 structure value representing the JMS message's header.
property	A G2 structure value representing the JMS message's property.
win	The G2 window used to display messages.

Return Value	Description
acknowledge-message	Returns true if the message is to be acknowledged in client_acknowledge mode; otherwise, returns false. See the description of jms-acknowledge-mode in jms-default-message-handler.

Description

This method of jms-interface handles all incoming JMS messages from the bridge. It can be customized for a specific G2 application.

jms-remove-all-messages

Removes all incoming messages from a buffer in the JMS interface object.

Synopsis

jms-remove-all-messages
    (*io*: jms-interface, *win*: g2-window)

Argument	Description
io	The jms-interface object used for connecting to the JMS provider.
win	The G2 window used to display messages.

Description

This method of jms-interface removes all incoming messages from a buffer of the JMS interface object.

The buffer is implemented as an attribute of the jms-interface object, jms-input-messages, whose value type is a G2 sequence. For details, see JMS Interface Attributes.

Error Handling

This section describes the procedure for responding to error conditions that occur in the G2 JMSLink bridge:

jms-default-bridge-error-handler

This procedure can be customized specifically for a G2 application.

jms-default-bridge-error-handler

Handles error conditions from the bridge.

Synopsis

jms-default-bridge-error-handler
(*io*: jms-interface, *msg*: value, *win*: ui-client-item)

Argument	Description
io	The jms-interface object used for connecting to the JMS provider.
msg	A G2 value representing the received JMS message.
win	The G2 window used to display messages.

Description

This method of jms-interface handles all error conditions reported from the bridge. It can be customized for a specific G2 application.

A

Message Properties

Describes how to use message properties to provide additional information about a message.

Introduction

This appendix describes how to use message properties for to provide more information about a message.

Message Properties

A `Message` object contains a built-in facility for supporting application-defined property values. This feature provides a mechanism for adding application-specific header fields to a message.

Properties allow an application, via message selectors, to have a JMS provider select or filter messages on its behalf, using application-specific criteria.

Property names must obey the rules for a message selector identifier.

Property values can be `boolean`, `byte`, `short`, `int`, `long`, `float`, `double`, and `String`.

Property values are set prior to sending a message. When a client receives a message, its properties are in read-only mode. If a client attempts to set properties

at this point, a `MessageNotWriteableException` is thrown. When `clearProperties` is called, the properties can be both read and written. Note that header fields are distinct from properties; header fields are never in read-only mode.

A property value might or might not duplicate a value in a message body. Although JMS does not define a policy for what should or should not be a property, application developers should note that JMS providers typically handle data in the message body more efficiently than it handles data in the message properties. For best performance, applications should use message properties only when they need to customize the message header. The primary reason for customizing the message header is to support customized message selection.

Message properties support the following conversion table. The specified types must be supported. The unmarked types must throw a `JMSException`. The conversions from String to primitive throws a runtime exception if the primitive's `valueOf` method does not accept the String as a valid representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	**byte**	**short**	**int**	**long**	**float**	**double**	**String**
boolean	✓							✓
byte		✓	✓	✓	✓			✓
short			✓	✓	✓			✓
int				✓	✓			✓
long					✓			✓
float						✓	✓	✓
double							✓	✓
String	✓	✓	✓	✓	✓	✓	✓	✓

In addition to the type-specific set/get methods for properties, JMS provides the `setObjectProperty` and `getObjectProperty` methods. These methods support the same set of property types, using primitive values as objects. Their purpose is to allow the decision of property type to be made at execution time rather than at compile time. These methods support the same property value conversions as described in the table above.

The `setObjectProperty` method accepts values of class `Boolean`, `Byte`, `Short`, `Integer`, `Long`, `Float`, `Double`, and `String`. An attempt to use any other class must throw a `JMSException`.

The `getObjectProperty` method only returns values of class `Boolean`, `Byte`, `Short`, `Integer`, `Long`, `Float`, `Double`, and `String`.

The order of property values is not defined. To iterate through the property values of a message, use `getPropertyNames` to retrieve a property name enumeration, then use the various property get methods to retrieve their values.

To delete the properties of a message, use the `clearProperties` method. This method leaves the message with an empty set of properties.

Getting a property value for a name that has not been set returns a null value. Only the `getStringProperty` and `getObjectProperty` methods can return a null value. Attempting to read a null value as a primitive type must be treated as calling the primitive's corresponding `valueOf(String)` conversion method with a null value.

The JMS API reserves the `JMSX` property name prefix for JMS-defined properties. The full set of these properties is defined in the Java Message Service specification. New JMS-defined properties might be added in later versions of the JMS API. Support for these properties is optional. The `String[] ConnectionMetaData.getJMSXPropertyNames` method returns the names of the JMSX properties supported by a connection.

A message selector can reference JMSX properties whether or not they are supported by a connection. If they are not present in a message, they are treated like any other absent property.

JMSX properties defined in the specification as "set by provider on send" are available to both the producer and the consumers of the message. JMSX properties defined in the specification as "set by provider on receive" are available only to the consumers.

`JMSXGroupID` and `JMSXGroupSeq` are standard properties that clients can use to group messages. All providers must support them. Unless specifically noted, the values and semantics of the JMSX properties are undefined.

The JMS API reserves the `JMS_vendor_name` property name prefix for provider-specific properties. Each provider defines its own value for `vendor_name`. A JMS provider uses this property to make its services available to a JMS client message.

The purpose of provider-specific properties is to provide special features needed to integrate JMS clients with provider-native clients in a single JMS application. They should not be used for messaging between JMS clients.

Provider Implementations of JMS Message Interfaces

The JMS API provides a set of message interfaces that define the JMS message model. It does not provide implementations of these interfaces.

Each JMS provider supplies a set of message factories with its `Session` object for creating instances of messages. This feature allows a provider to use message implementations tailored to its specific needs.

A provider must be prepared to accept message implementations that are not its own. They may not be handled as efficiently as its own implementation; however, they must be handled.

Note the following exceptions when a provider is handling a foreign message implementation. If the foreign message implementation contains a `JMSReplyTo` header field that is set to a foreign destination implementation, the provider is not required to handle or preserve the value of this header field.

B

Message Selectors

Describes how to use message selectors filter messages received from a topic or queue.

Introduction

A JMS message selector allows a client to specify the messages it is interested in by using header field and property references. Only messages whose header and property values match the selector are delivered. The `MessageConsumer` determines what it means for a message not to be delivered.

Message selectors cannot reference message body values.

A message selector matches a message if the selector evaluates to true when the message header field values and property values are substituted for their corresponding identifiers in the selector.

A message selector is a `String` whose syntax is based on a subset of the SQL92 conditional expression syntax. If the value of a message selector is an empty string, the value is treated as null and indicates that there is no message selector for the message consumer.

The order of evaluation of a message selector is from left to right within precedence level. Use parentheses to change this order.

Predefined selector literals and operator names appear below in uppercase; however, they are case insensitive.

Message Selectors

A selector can contain:

- Literals:
  - A string literal is enclosed in single quotes, with a single quote represented by doubled single quote, for example, `'literal'` and `'literal''s'`. Like string literals in the Java programming language, these use the Unicode character encoding.
  - An exact numeric literal is a numeric value without a decimal point, such as `57`, `-957`, and `+62`; numbers in the range of `long` are supported. Exact numeric literals use the integer literal syntax of the Java programming language.
  - An approximate numeric literal is a numeric value in scientific notation, such as `7E3` and `-57.9E2`, or a numeric value with a decimal, such as `7.`, `-95.7`, and `+6.2`; numbers in the range of double are supported. Approximate literals use the floating-point literal syntax of the Java programming language.
  - The boolean literals `TRUE` and `FALSE`.
- Identifiers:
  - An identifier is an unlimited-length sequence of letters and digits, the first of which must be a letter. A letter is any character for which the method `Character.isJavaLetter` returns true. This includes `'_'` and `'$'`. A letter or digit is any character for which the method `Character.isJavaLetterOrDigit` returns true.
  - Identifiers cannot be the names `NULL`, `TRUE`, and `FALSE`.
  - Identifiers cannot be `NOT`, `AND`, `OR`, `BETWEEN`, `LIKE`, `IN`, `IS`, or `ESCAPE`.
  - Identifiers are either header field references or property references. The type of a property value in a message selector corresponds to the type used to set the property. If a property that does not exist in a message is referenced, its value is `NULL`.
  - The conversions that apply to the get methods for properties do not apply when a property is used in a message selector expression. For example, suppose you set a property as a string value, as in the following:

```
myMessage.setStringProperty("NumberOfOrders", "2");
```

The following expression in a message selector would evaluate to false, because a string cannot be used in an arithmetic expression:

```
"NumberOfOrders > 1"
```

  - Identifiers are case-sensitive.
  - Message header field references are restricted to `JMSDeliveryMode`, `JMSPriority`, `JMSMessageID`, `JMSTimestamp`, `JMSCorrelationID`, and `JMSType`. `JMSMessageID`, `JMSCorrelationID`, and `JMSType` values may be null and if so are treated as a `NULL` value.
  - Any name beginning with `'JMSX'` is a JMS-defined property name.
  - Any name beginning with `'JMS_'` is a provider-specific property name.
  - Any name that does not begin with `'JMS'` is an application-specific property name.
- White space is the same as that defined for the Java programming language: space, horizontal tab, form feed, and line terminator.
- Expressions:
  - A selector is a conditional expression. A selector that evaluates to true matches. A selector that evaluates to false or unknown does not match.
  - Arithmetic expressions are composed of themselves, arithmetic operations, identifiers (whose value is treated as a numeric literal), and numeric literals.
  - Conditional expressions are composed of themselves, comparison operations, and logical operations.
- Standard bracketing `()` for ordering expression evaluation is supported.
- Logical operators in precedence order: `NOT`, `AND`, `OR`
- Comparison operators: `=`, `>`, `>=`, `<`, `<=`, `<>` (not equal)
  - Only like type values can be compared. One exception is that it is valid to compare exact numeric values and approximate numeric values. The type conversion required is defined by the rules of numeric promotion in the Java programming language. If the comparison of non-like type values is attempted, the value of the operation is false. If either of the type values evaluates to `NULL`, the value of the expression is unknown.
  - String and boolean comparison is restricted to = and <>. Two strings are equal if and only if they contain the same sequence of characters.

- Arithmetic operators in precedence order:
  - `+, -` (unary)
  - `*, /` (multiplication and division)
  - `+, -` (addition and subtraction)
  - Arithmetic operations must use numeric promotion in the Java programming language.
- *arithmetic-expr1* `[NOT] BETWEEN` *arithmetic-expr2* `AND` *arithmetic-expr3* (comparison operator)
  - `"age BETWEEN 15 AND 19"` is equivalent to `"age >= 15 AND age <= 19"`
  - `"age NOT BETWEEN 15 AND 19"` is equivalent to `"age < 15 OR age > 19"`
- `identifier [NOT] IN (`*string-literal1*`,` *string-literal2*`,...)` (comparison operator where `identifier` has a `String` or `NULL` value)
  - `"Country IN (' UK', 'US', 'France')"` is true for `'UK'` and false for `'Peru'`. It is equivalent to the expression `"(Country = ' UK') OR (Country = ' US') OR (Country = ' France')"`
  - `"Country NOT IN (' UK', 'US', 'France')"` is false for `'UK'` and true for `'Peru'`; it is equivalent to the expression `"NOT ((Country = ' UK') OR (Country = ' US') OR (Country = ' France'))"`
  - If identifier of an `IN` or `NOT IN` operation is `NULL`, the value of the operation is unknown.
- `identifier [NOT] LIKE` *pattern-value* `[ESCAPE` *escape-character*`]` (comparison operator, where identifier has a `String` value; *pattern-value* is a string literal where `'_'` stands for any single character; `'%'` stands for any sequence of characters, including the empty sequence; and all other characters stand for themselves. The optional *escape-character* is a single-character string literal whose character is used to escape the special meaning of the `'_'` and `'%'` in *pattern-value*)
  - `"phone LIKE '12%3'"` is true for `'123'` or `'12993'` and false for `'1234'`
  - `"word LIKE 'l_se'"` is true for `'lose'` and false for `'loose'`
  - `"underscored LIKE '\_%' ESCAPE '\'"` is true for `'_foo'` and false for `'bar'`
  - `"phone NOT LIKE '12%3'"` is false for `'123'` or `'12993'` and true for `'1234'`
  - If identifier of a `LIKE` or `NOT LIKE` operation is `NULL`, the value of the operation is unknown.

- `identifier IS NULL` (comparison operator that tests for a null header field value or a missing property value)
  - `"prop_name IS NULL"`
- `identifier IS NOT NULL` (comparison operator that tests for the existence of a non-null header field value or a property value)
  - `"prop_name IS NOT NULL"`

JMS providers are required to verify the syntactic correctness of a message selector at the time it is presented. A method that provides a syntactically incorrect selector must result in a `JMSException`.

The following message selector selects messages with a message type of car and color of blue and weight greater than 2500 pounds:

```
"JMSType = 'car' AND color = 'blue' AND weight > 2500"
```

Null Values

As noted above, property values may be `NULL`. The evaluation of selector expressions containing `NULL` values is defined by `SQL92 NULL` semantics. A brief description of these semantics is provided here.

SQL treats a `NULL` value as unknown. Comparison or arithmetic with an unknown value always yields an unknown value.

The `IS NULL` and `IS NOT NULL` operators convert an unknown value into the respective `TRUE` and `FALSE` values.

The boolean operators use three-valued logic as defined by the following tables.

The definition of the AND operator:

AND	T	F	U
T	T	F	U
F	F	F	F
U	U	F	U

The definition of the OR operator:

OR	T	F	U
T	T	T	T
F	T	F	U
U	T	U	U

The definition of the NOT operator:

OR	
T	F
F	T
U	T

Special Notes

When used in a message selector, the `JMSDeliveryMode` header field is treated as having the values `'PERSISTENT'` and `'NON_PERSISTENT'`.

Date and time values should use the standard long millisecond value. When a date or time literal is included in a message selector, it should be an integer literal for a millisecond value. The standard way to produce millisecond values is to use `java.util.Calendar`.

Although SQL supports fixed decimal comparison and arithmetic, JMS message selectors do not. This is the reason for restricting exact numeric literals to those without a decimal (and the addition of numerics with a decimal as an alternate representation for approximate numeric values).

SQL comments are not supported.

Index

O

P

S