Driver for Java Messaging Service Implementation Guide

Identity Manager 4.0.1

December 07, 2011

Novell, Inc.
1800 South Novell Place
Provo, UT 84606
U.S.A.
www.novell.com

Online Documentation: To access the online documentation for this and other Novell products, and to get updates, see Novell Documentation (http://www.novell.com/documentation/).

Contents

About This Guide

This guide explains how to install and configure the Identity Manager Driver for Java Messaging Service (JMS).

Audience

This guide is intended for developers and administrators using Identity Manager and the JMS driver.

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation.

Documentation Updates

For the most recent version of this guide, visit the Identity Manager 4.0.1 Driver Documentation Web site (http://www.novell.com/documentation/idm401drivers).

Additional Documentation

For documentation on Identity Manager and other drivers, see the Identity Manager 4.0.1 Documentation Web site (http://www.novell.com/documentation/idm401).

1 Understanding the JMS Driver

The Identity Manager Driver for Java Messaging Service (JMS), hereafter referred to as the *JMS driver* or simply the *driver*, provides Identity Manager integration with various applications that are used for messaging. The driver is JMS-generic and does not target any specific application or messaging provider. It supports all versions of the JMS API defined by Sun Microsystems.

The following sections introduce concepts you should understand before using the driver:

1.1 Supported JMS Vendors and Versions

The driver supports the following vendors and versions:

- JBossMQ v4.2.2
- JBoss Messaging 1.3.0
- IBM WebSphere MQ v6.x and MQ v7.x
- SonicMQ v7.*x*
- TIBCO EMS v4 and v5

The driver uses two main specifications of JMS, 1.0.2b and 1.1.

1.2 Key Terminology

The following terms are used throughout this document:

- **JMS:** Java Messaging Service. The driver uses two main specifications of JMS, 1.0.2b and 1.1.
- **JNDI:** Java Naming and Directory Interface. JNDI is used to look up, connect, and authenticate to message brokers.
- **Message Broker:** The server that handles message interchange between messaging clients.
- **Messaging Client:** Messaging clients produce and consume messages. The driver is a messaging client, and so are third-party applications.
- **Destination:** The abstract term for a topic or a queue.

- **Session:** A per-thread connection. Each thread creates one or more sessions from a connection to communicate with the message broker.
- **Persistence:** Persistence guarantees that a message is delivered only once; this can be controlled on a per-message basis. Message brokers usually support persistent storage via an underlying database. This is sometimes referred to as stable storage.
- **Durability:** The message broker stores messages for a message receiver when the receiver is inactive or disconnected.
- **Acknowledgement:** When transactions are not being used, a client acknowledges receipt of a message to the message broker in *CLIENT_ACKNOWLEDGE* mode. In this mode, the client must explicitly acknowledge receipt of one or more messages by committing the current transaction. By rolling back the current transaction, all received messages are re-delivered (or set to *retry*, in Identity Manager terminology.)

1.3 JMS Messaging Models

The driver supports two messaging models: Point-to-Point messaging and Publish/Subscribe messaging.

- Section 1.3.1, "Point-to-Point Messaging," on page 8
- Section 1.3.2, "Publish/Subscribe Messaging," on page 9

The JMS API also uses abstract names. To better understand how these abstract names correspond to model terminology, see the table below.

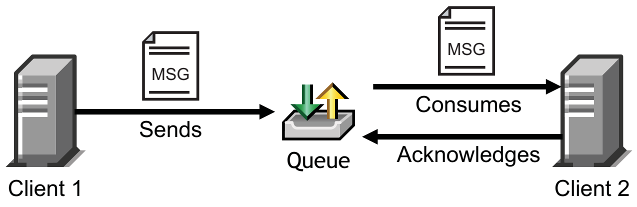
Abstract Terminology	Point-to-Point Terminology	Publish/Subscribe Terminology
Destination	Queue	Topic
Sender (or Producer)	Sender	Publisher
Receiver (or Consumer)	Receiver	Subscriber

1.3.1 Point-to-Point Messaging

Point-to-Point messaging is used when one client needs to send a message to another client. As illustrated in Figure 1-1, Client 1 is the sender and Client 2 is the receiver. The queue receives messages, while the message broker receives any acknowledgements.

In Point-to-Point messaging there is a one-to-one relationship between senders and receivers. You configure durability on the broker side.

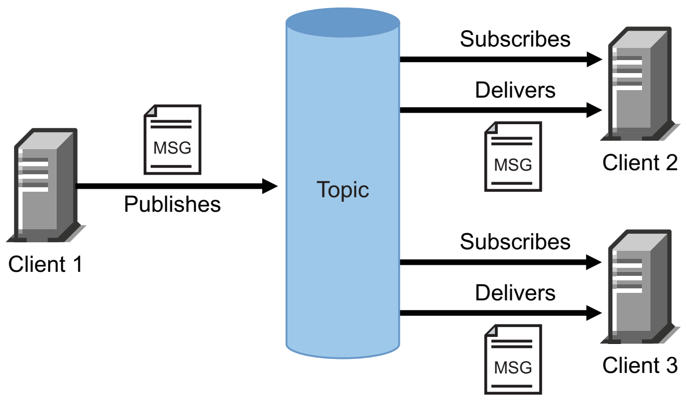
Figure 1-1 *Point-to-Point Messaging*



1.3.2 Publish/Subscribe Messaging

Publish/Subscribe messaging is used when multiple applications need to receive the same messages. Multiple publishers can send messages to a topic, and all subscribers to that topic receive all the messages sent to that topic. This model is useful when a group of applications want to notify each other of a particular event.Publish/Subscribe messaging allows for one-to-many or many-to-many implementations. Durability is configured on either the client side or the broker side.

Figure 1-2 *Publish/Subscribe Messaging*



1.4 JMS Messages

The following sections contain information about JMS message structures and message types, as well as examples for each.

- Section 1.4.1, "Message Structure," on page 9
- Section 1.4.2, "Message Types," on page 10

1.4.1 Message Structure

JMS messages consist of metadata (comprised of headers and properties) and message data (a body). In order to make message metadata accessible to policy processing, messages sent to the driver can be wrapped in an envelope, and messages received by the driver and sent to the Metadirectory engine are also wrapped in an envelope. All message envelope elements and special attributes must have a namespace prefix bound to *urn:idm:jms*. For consistency, the namespace prefix *jms* is used throughout this document. The root message envelope element *jms:message* must be a child of the XDS input/output elements.

Example JMS Message Envelope

```
<jms:message xmlns:jms="urn:idm:jms">
<jms:headers>
      <!-- standard JMS headers start with "JMS" -->
      <!-- client-assignable headers -->
      <jms:header jms:name="JMSDeliveryMode"/>
      <jms:header jms:name="JMSExpiration"/>
      <jms:header jms:name="JMSPriority"/
      <jms:header jms:name="JMSReplyTo"/>
      <jms:header jms:name="JMSCorrelationID"/>
      <jms:header jms:name="JMSType"/>
   </jms:headers>
   <jms:properties>
      <!-- standard JMS properties start with "JMSX" -->
      <jms:property jms:name="JMSXUserID"/>
      <jms:property jms:name="JMSXAppID"/>
      <jms:property jms:name="JMSXProducerTXID"/>
      <jms:property jms:name="JMSXConsumerTXID"/>
      <jms:property jms:name="JMSXRcvTimestamp"/>
      <jms:property jms:name="JMSXDeliveryCount"/>
      <jms:property jms:name="JMSXState"/>
      <jms:property jms:name="JMSXGroupID"/>
      <jms:property jms:name="JMSXGroupSeq"/>
      <!-- provider-specific properties start with "JMS_" -->
      <!-- application-specific properties start with anything else -->
   </jms:properties>
  <jms:body/>
</jms:message>
```

1.4.2 Message Types

Message type refers to how a message is sent, not necessarily what its content is. For example, a text message can be sent as text or bytes. The driver supports both text and bytes messages.

Example Text Message

```
<jms:message xmlns:jms="urn:idm:jms">
   <jms:properties>
      <!-- send message as text -->
      <jms:property name="Novell_IDM_MessageType">text</jms:property>
   </jms:properties>
   <jms:body>content</jms:body>
</jms:message>
```

Example Bytes Message

```
<jms:message xmlns:jms="urn:idm:jms">
   <jms:properties>
      <!-- send message as bytes -->
      <jms:property name="Novell_IDM_MessageType">bytes</jms:property>
   </jms:properties>
   <jms:body>content</jms:body>
</jms:message>
```

1.5 How Subscriber and Publisher Channels Work

The following sections contain information about how the Subscriber and Publisher channels work with the JMS driver. This driver functions differently than traditional Identity Manager drivers, so it's important to review this information.

- Section 1.5.1, "Subscriber Channel," on page 11
- Section 1.5.2, "Publisher Channel," on page 12

1.5.1 Subscriber Channel

The Subscriber channel is capable of sending messages to (and optionally receiving messages from) multiple destinations on a single broker. Multi-broker support is not yet implemented. As a side effect of sending a JMS message, the Subscriber channel can receive a response within a specified timeout interval. Message routing and RPC (Remote Procedure Call) emulation are achieved via three special attributes: *jms:send-to*, *jms:receive-from* and *jms:receive-timeout*.

Example Special Attributes

```
<jms:message xmlns:jms="urn:idm:jms"
               jms:send-to="queueA"
               jms:receive-from="queueB"
             jms:receive-timeout-seconds="10"/>
```

These attributes can be used on a *jms:message* envelope tag or any XDS command that is a child of input/output elements. The destination names used in these parameters can either be the JNDI (Java Naming and Directory Interface) name or the unique Identity Manager identifier assigned to the destination in the driver configuration. By default, the Subscriber sends messages to the first-defined send destination and does not wait for a message response (meaning that the message receipt is assumed to be asynchronous).

By using a JMS message envelope, it is possible to override headers/properties or add vendor-specific properties or application properties.

```
<jms:message xmlns:jms="urn:idm:jms">
   <jms:headers>
   <!-- override standard headers -->
       <jms:header jms:name="JMSType">type</jms:header>
       <jms:header jms:name="JMSCorrelationID">blah</jms:header>
      <jms:header jms:name="JMSDeliveryMode">non-persistent</jms:header>
      <jms:header jms:name="JMSExpiration">10000</jms:header>
      <jms:header jms:name="JMSPriority">9</jms:header>
      <jms:header jms:name="JMSReplyTo">A</jms:header>
   </jms:headers>
   <jms:properties>
      <!-- add/override vendor-specific properties -->
      <jms:property jms:name="JMS_IBM_Format">MQSTR</jms:property>
      <!-- add/override application properties -->
      <jms:property jms:name="Novell_IDM_MessageType">bytes</jms:property>
      <jms:property jms:name="Novell_IDM_ContentType">xml</jms:property>
      <jms:property jms:name="Novell_IDM_CharEncoding">UTF-8</jms:property>
   </jms:properties>
   <jms:body>text</jms:body>
</jms:message>
```

1.5.2 Publisher Channel

The Publisher channel is essentially a Subscriber channel (you can send messages to a broker as a side effect of publishing messages—including heartbeat documents—and wait for a response) with the added ability to periodically monitor specified destinations to receive messages for publication.

You can configure the Publisher channel to monitor an unlimited number of destinations on a single message broker bounded only by certain practical considerations. Having too many monitored destinations can significantly slow down rendering of driver parameters in iManager or Designer, as currently implemented. Having too many destinations can result in decreased performance because all destinations are being monitored by a single thread. Furthermore, there is a finite amount of space available for storing a driver's configuration (64 KB).

The Publisher channel polls each monitored destination in a round-robin fashion, starting with the first declared destination. A polling cycle ends when all monitored destinations fail to return messages, at which time the Publisher sleeps for the specified polling interval until it's time to start a new polling cycle.

The Publisher channel receives messages in a synchronous fashion as opposed to an asynchronous one. The main reason for this is to prevent client overrun—when the message broker feeds messages to the driver faster than it can process them—that can lead to memory exhaustion.

1.6 Support for Standard Driver Features

The following sections provide information about how the JMS driver supports these standard driver features:

- Section 1.6.1, "Local Platforms," on page 12
- Section 1.6.2, "Remote Platforms," on page 12
- Section 1.6.3, "Entitlements," on page 13
- Section 1.6.4, "Password Synchronization Support," on page 13
- Section 1.6.5, "Information Synchronized," on page 13

1.6.1 Local Platforms

A local installation is an installation of the driver on the same server as the Metadirectory engine, Identity Vault, and JMS vendor application. Both systems that the driver needs to communicate with (Metadirectory engine and JMS) are local to the driver.

The JMS driver can be installed on the same operating systems that are supported by the Metadirectory server. For information about the operating systems supported by the Metadirectory server, see "System Requirements" in the *Identity Manager 4.0.1 Integrated Installation Guide*.

1.6.2 Remote Platforms

The JMS driver can use the Remote Loader service. The Remote Loader service for the JMS driver can be installed on any of the Identity Manager supported platforms.

For more information about installing the Remote Loader services, see "Installing Identity Manager" in the *Identity Manager 4.0.1 Integrated Installation Guide*.

1.6.3 Entitlements

The JMS driver does not have Entitlement functionality defined in its basic configuration files. The driver does support entitlements, if there are policies created for the driver to consume.

1.6.4 Password Synchronization Support

The basic configuration files for the JMS driver do not include policies for synchronizing passwords.

1.6.5 Information Synchronized

The JMS driver synchronizes any messaging format you want. By default, the driver is set up with a Loopback driver configuration.

1.7 Additional Resources

For more information about JMS and messaging models, see the following Web sites:

- Sun's Developer Network FAQ on the JMS API (http://java.sun.com/products/jms/faq.html)
- Getting Started with JMS (http://java.sun.com/developer/technicalArticles/Ecommerce/jms/index.html)
- JMS Tutorial (http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html)
- JMS Specifications (1.0.2b and 1.1) (http://java.sun.com/products/jms/docs.html)

2 Installing the Driver Files

By default, the JMS driver files are installed on the Metadirectory server at the same time as the Metadirectory engine. The installation program extends the Identity Vault's schema and installs both the driver shim and the driver configuration files. It does not create the driver in the Identity Vault (see Chapter 4, "Creating a New Driver," on page 27) or upgrade an existing driver's configuration (see Chapter 5, "Upgrading an Existing Driver," on page 33).

The JMS driver must be located on the same server as your JMS vendor. If the driver is not on that server, you have the following options:

- On a local machine: Install the JMS driver files on the Metadirectory server and connect to the JMS server by using the JMS Broker URL (Connection Properties). See the instructions in "Installing Identity Manager" in the *Identity Manager 4.0.1 Integrated Installation Guide*.
- On a remote machine: Install the JMS driver files on the Remote Loader. See the instructions in "Installing Identity Manager" in the *Identity Manager 4.0.1 Integrated Installation Guide*.

3 Configuring Messaging Vendors

The following sections provide information about configuring your JMS vendor to work with the JMS driver:

3.1 Installing IBM WebSphere MQ on Win32

As part of installing WebSphere for the driver, you should complete the following tasks consecutively. These instructions are for Windows, but you can follow the same procedure for other platforms.

3.1.1 Placing Prerequisite Jar Files and Scripts

1. On your messaging server, locate the following jar files:
   - `com.ibm.mq.jar`
   - `com.ibm.mqjms.jar`
   - `connect.jar`
   - `dhbcore.jar`
   - `jta.jar`
   - `fscontext.jar`
   - `jndi.jar`
2. Copy the jar files to the Identity Manager server.

   The following table identifies where to place jar files on an Identity Management server, by platform.

Platform	Directory Path
Windows	Local installation: `C:\Novell\IdentityManager\NDS\lib` Remote installation: `C:\Novell\IdentityManager\RemoteLoader\lib`
Linux/UNIX	Local installation: `/usr/lib/dirxml/classes` (pre-eDirectory 8.8) or `opt/novell/eDirectory/lib/dirxml/classes` (eDirectory 8.8) Remote installation: `/usr/lib/dirxml/classes` (pre-eDirectory 8.8) or `/opt/novell/eDirectory/lib/dirxml/classes` (eDirectory 8.8)

3 Locate where you installed the installation script during the JMS driver installation. The following table indicates the default directories where scripts are installed, by platform.

Platform	Directory Path
Windows	`C:\Novell\IdentityManager\NDS\DirXMLUtilities\jms\webmq`
Linux\UNIX	`install-dir/lib/dirxml/rules/jms\webmq`

4 Copy the script to your messaging server.

5 If necessary, restart your eDirectory server.

3.1.2 Creating a Server-Connection Channel and Queues

1 From the command line, change directories to `Program Files\IBM\WebSphere MQ\Java\bin`.

2 From the command line, execute the following command:

```
runmqsc QM < idm_mq_install.mqsc
```

This file is provided only as an example; you might need to customize the content .

3 Continue with Section 3.1.3, "Starting the Publish/Subscriber Broker," on page 18.

3.1.3 Starting the Publish/Subscriber Broker

1 From the command line, execute the following command:

```
strmqbrk -m QM
```

You should see a message indicating that the broker is running.

2 Continue with Section 3.1.4, "Installing System Queues Necessary for Publish/Subscribe," on page 18.

3.1.4 Installing System Queues Necessary for Publish/Subscribe

1 From the command line, execute the following command:

```
runmqsc QM < MQJMS_PSQ.mqsc
```

You should see some tracing, indicating successful queue creation.

NOTE: If you don't enter this command, you might see the following error: "`MQJMS1111: JMS 1.1 The required Queues/Publish Subscribe services are not set up {0} error.`"

2 Continue with Section 3.1.5, "Creating a User Account," on page 19.

3.1.5 Creating a User Account

- "Creating a User" on page 19
- "Making the User a Member of the mqm Group" on page 19

Creating a User

1 Click *Start* > *Programs* > *Administrative Tools* > *Computer Management*.

2 Expand the *Local Users and Groups* subtree.

3 Right-click the *Users* folder, then select *New User*.

4 Specify a username. The scripts referenced in these instructions assume *idm*.

5 Specify a password. The scripts referenced in these instructions assume *novell*.

6 Deselect the *User must change password at next login* check box.

7 Click the *Create* button.

8 Click the *Close* button.

9 Continue with "Making the User a Member of the mqm Group" on page 19.

Making the User a Member of the mqm Group

1 Right-click the newly created user, then click *Properties*.

2 Select the *Member Of* tab.

3 Select the mqm group.

4 Click *Add*.

5 Click *OK* twice.

6 Continue with Section 3.1.6, "Setting Up JMS," on page 19.

3.1.6 Setting Up JMS

1 Edit the `Program Files\IBM\WebSphere MQ\Java\bin\JMSAdmin.bat` file

```
@echo off
::add this line at the beginning of the file
setlocal

::add the following line before call to java
set    JRE_PATH=C:\Program Files\IBM\WebSphere MQ\gskit\jre

::replace call to Java
"%JRE_PATH%\bin\java" -cp "%CLASSPATH%"
-DMQJMS_INSTALL="%MQ_JAVA_INSTALL_PATH%" -
DMQJMS_LOG_DIR="%MQ_JAVA_DATA_PATH%"\log -
DMQJMS_TRACE_DIR="%MQ_JAVA_DATA_PATH%"\errors -
DMQJMS_INSTALL_PATH="%MQ_JAVA_INSTALL_PATH%" com.ibm.mq.jms.admin.JMSAdmin %1
%2 %3 %4 %5

::add this line at end of file
endlocal
```

2 Edit the `Program Files\IBM\WebSphere MQ\Java\bin\JMSAdmin.config` file:

```
# comment out all of the INITIAL_CONTEXT_FACTORY lines using
# comment char "#" and add this line:
INITIAL_CONTEXT_FACTORY=com.sun.jndi.fscontext.RefFSContextFactory
# comment out all PROVIDER_URL lines and add this one:
PROVIDER_URL=file://<hostmname>:<port>/<path of binding file>
```

3 Locate where you installed the installation script during the driver installation. The following table indicates the default directories where scripts are installed by platform.

Platform	Directory Path
Windows	`C:\Novell\IdentityManager\NDS\DirXMLUtilities\jms\webmq`
Linux/UNIX	`install-dir/lib/dirxml/rules/jms/webmq`

4 Copy the following scripts to the `Program Files\IBM\WebSphere MQ\Java\bin` directory on your messaging server:

- `idm_jms_install.scp`
- `idm_jms_uninstall.scp`
- `idm_mq_install.mqsc`
- `idm_mq_uninstall.mqsc`
- `install.bat`
- `uninstall.bat`

5 Edit the `idm_jms_install.scp` configuration file with definitions for each queue that is defined to connect with the Identity Manager JMS driver. For example, if the queue manager name is `QM1`, the publisher queue name is `PQ1`, and the subscriber queue is `SQ1`, then the definitions in the `idm_jms_install.scp` configuration file should be as follows:

```
-- Queue connection factory definition
define qcf(QueueConnectionFactory) +
       qmgr(QM1) +
       tran(CLIENT) +
       host(hostname) +
       port(1414)

-- Publisher Queue Definition---
define q(PQ1) +
       qmgr(QM1) +
       queue(PQ1)

-- Subscriber Queue Definition---
define q(SQ2) +
       qmgr(QM1) +
       queue(SQ2)

    -- q is the jndi name used in the JMS driver configuration.
    -- qmgr is the queue manager name.
    -- queue is the queue name.
    -- host is the IP address or host Name of the IBM Websphere MQ server.
    -- port is the queue manager listener port.
```

6. Update the listener port in `idm_mq_install.mqsc`.
7. From the command line, change directories to `Program Files\IBM\WebSphere MQ\Java\bin`.
8. From the command line, execute the following command:

   ```
   JMSAdmin.bat -v < idm_jms_install.scp
   ```

   This file is provided as an example only; you might need to customize the content.
9. From the command line, manually start the publish/subscribe broker by executing the following command:

   ```
   Program Files\IBM\WebSphere MQ\bin\strmqbrk.exe.
   ```
10. From the command line, ensure that the publish/subscribe broker is configured correctly by executing the following command:

    ```
    Program Files\IBM\WebSphere MQ\Java\PSIVTRun.bat -nojndi -t
    ```
11. Make sure the `.bindings` file resides in the correct location.

    The `.bindings` file is generated during the WebSphere MQ configuration. When you run the `JMSAdmin.bat -v idm_jms_install.scp` command, the `.bindings` file is generated under the path specified in the `JMSAdmin.config` file.

    If the driver, WebSphere MQ, Metadirectory engine, and Identity Vault are all on the same server, make sure the `.bindings` file resides in the location specified by the PROVIDER_URL option for the driver configuration (see PROVIDER_URL).

    If the driver and WebSphere MQ are on one server and the Metadirectory engine and Identity Vault are on another server (a Metadirectory server), copy the `.bindings` file to the Metadirectory server and make sure the PROVIDER_URL includes the correct path to the file. If multiple Metadirectory servers connect to the WebSphere MQ server, copy the `.bindings` file to the PROVIDER_URL path on each Metadirectory server.

3.2 Installing on JBoss Messaging

As part of installing JBoss for the driver, you should copy the jar files as indicated below. The instructions assume that JBoss already has the default queues and topics available. For information on installing and configuring JBoss Messaging, refer to the JBoss User Guide (http://www.jboss.org/file-access/default/members/jbossmessaging/freezone/docs/userguide-1.3.0.GA/html/index.html).

1. On your messaging server, locate the following jar files:
   - `concurrent.jar`
   - `connector.jar`
   - `javaassist.jar`
   - `jboss-aop-jdk50.jar`
   - `jboss-aop-jdk50-client.jar`
   - `jboss-common-client.jar`
   - `jboss-messaging.jar`
   - `jboss-messaging-client.jar`
   - `jboss-remoting.jar`
   - `jboss-system-client.jar`
   - `jnp-client.jar`
   - `trove.jar`
2. Copy the jar files to the Identity Manager server.

   The following table identifies where to place jar files on an Identity Management server, by platform.

Platform	Directory Path
Windows	Local installation: `novell\NDS\lib` Remote installation: `novell\RemoteLoader\lib`
Linux/UNIX	Local installation: `/usr/lib/dirxml/classes` (pre-eDirectory 8.8) or `opt/novell/eDirectory/lib/dirxml/classes` (eDirectory 8.8) Remote installation: `/usr/lib/dirxml/classes` (pre-eDirectory 8.8) or `/opt/novell/eDirectory/lib/dirxml/classes` (eDirectory 8.8)

3. If necessary, restart your eDirectory server.

3.3 Installing on SonicMQ

As part of installing SonicMQ for the driver, you should complete the following tasks consecutively. These instructions are for Linux, but you can follow the same procedure for other platforms.

- Section 3.3.1, "Locating Prerequisite Jar Files," on page 23
- Section 3.3.2, "Running Scripts to Configure the Messaging System," on page 23

3.3.1 Locating Prerequisite Jar Files

1 On your messaging server, locate the following jar files:

- `mfcontext.jar`
- `sonic_ASPI.jar`
- `sonic_Channel.jar`
- `sonic_Client.jar`
- `sonic_Crypto.jar`
- `sonic_Selector.jar`
- `sonic_SF.jar`
- `sonic_SSL.jar`
- `sonic_XA.jar`
- `sonic_XMessage.jar`

2 Copy the jar files to the Identity Manager server.

The following table identifies where to place jar files on an Identity Management server, by platform.

Platform	Directory Path
Windows	Local installation: `novell\NDS\lib` Remote installation: `novell\RemoteLoader\lib`
Linux/UNIX	Local installation: `/usr/lib/dirxml/classes` (pre-eDirectory 8.8) or `opt/novell/eDirectory/lib/dirxml/classes` (eDirectory 8.8) Remote installation: `/usr/lib/dirxml/classes` (pre-eDirectory 8.8) or `/opt/novell/eDirectory/lib/dirxml/classes` (eDirectory 8.8)

3 If necessary, restart your eDirectory server.

4 Continue with Section 3.3.2, "Running Scripts to Configure the Messaging System," on page 23.

3.3.2 Running Scripts to Configure the Messaging System

Use the following instructions to locate and run the scripts to configure your message system.

1 Locate where you installed the installation script (`idm_jms_install.cli`) during the JMS driver installation. The following table indicates the default directories where scripts are installed, by platform.

Platform	Directory Path
Windows	`C:\Novell\IdentityManager\NDS\DirXMLUtilities\jms\sonic`
Linux\UNIX	`install-dir/lib/dirxml/rules/jms\sonic`

2 Copy the script to your messaging server.

3 Follow the instructions provided in the script.

3.4 Installing on TIBCO EMS

As part of installing TIBCO for the driver, you should complete the following tasks consecutively. These instructions are for Linux and Windows.

- Section 3.4.1, "Locating Prerequisite Client Jar Files," on page 24
- Section 3.4.2, "Running Scripts to Configure the Messaging System," on page 24

3.4.1 Locating Prerequisite Client Jar Files

1 On your messaging server, locate the following jar files:
   - `tibjms.jar`
   - `tibcrypt.jar`

2 The following table identifies where to place jar files on a TIBCO server, by platform:

Platform	Directory Path
Windows	`C:tibco\ems\clients\java`
Linux	`/opt/tibco/ems/clients/java`

3 Copy the jar files to the Identity Manager server.

   The following table identifies where to place jar files on an Identity Management server, by platform:

Platform	Directory Path
Windows	Local installation: `novell\NDS\lib` Remote installation: `novell\RemoteLoader\lib`
Linux/UNIX	Local installation: `/usr/lib/dirxml/classes` (pre-eDirectory 8.8) or `opt/novell/eDirectory/lib/dirxml/classes` (eDirectory 8.8) Remote installation: `/usr/lib/dirxml/classes` (pre-eDirectory 8.8) or `/opt/novell/eDirectory/lib/dirxml/classes` (eDirectory 8.8)

4 If necessary, restart your eDirectory server.

5 Continue with Section 3.4.2, "Running Scripts to Configure the Messaging System," on page 24.

3.4.2 Running Scripts to Configure the Messaging System

Use the following instructions to locate and run the scripts to configure your message system:

1 Locate where you installed the installation script (`idm_jms_install.tib`) during the driver installation. The following table indicates the default directories where scripts are installed, by platform:

Platform	Directory Path
Windows	`C:\Novell\IdentityManager\NDS\DirXMLUtilities\jms\tibco_ems`

Platform	Directory Path
Linux\UNIX	`install-dir/lib/dirxml/rules/jms\tibco_ems`

2 Copy the `idm_jms_install.tib` and `idm_jms_uninstall.tib` scripts to your messaging server. The following table indicates the location where you should copy the scripts to on your messaging server, by platform.

Platform	Directory Path
Windows	`C:\tibco\ems\bin`
Linux/UNIX	`/opt/tibco/ems/bin`

3 Update the IP address and port number of the connection factory in the `idm_jms_install.tib` script.

4 Change directories on the messaging server to run the tibjmsadmin utility. The following table indicates where the tibjmsadmin utility is installed, by platform.

Platform	Directory Path
Windows	`C:\tibco\ems\bin`
Linux/UNIX	`/opt/tibco/ems/bin`

5 To run the installation script, enter the following at the command line prompt:

```
>tibjmsadmin -script idm_jms_install.tib
```

4 Creating a New Driver

After the JMS driver files are installed on the server where you want to run the driver (see Chapter 2, "Installing the Driver Files," on page 15) and you've configured the JMS system (see Chapter 3, "Configuring Messaging Vendors," on page 17), you can create the driver in the Identity Vault. You do so by installing the driver packages and then modifying the driver configuration to suit your environment.

- Section 4.1, "Creating the Driver in Designer," on page 27
- Section 4.2, "Creating the Driver in iManager," on page 30
- Section 4.3, "Activating the Driver," on page 31

4.1 Creating the Driver in Designer

You create the JMS driver by installing the driver packages and then modifying the configuration to suit your environment. After you create and configure the driver, you need to deploy it to the Identity Vault and start it.

- Section 4.1.1, "Importing the Current Driver Packages," on page 27
- Section 4.1.2, "Installing the Driver Packages," on page 28
- Section 4.1.3, "Configuring the Driver," on page 29
- Section 4.1.4, "Deploying the Driver," on page 29
- Section 4.1.5, "Starting the Driver," on page 30

4.1.1 Importing the Current Driver Packages

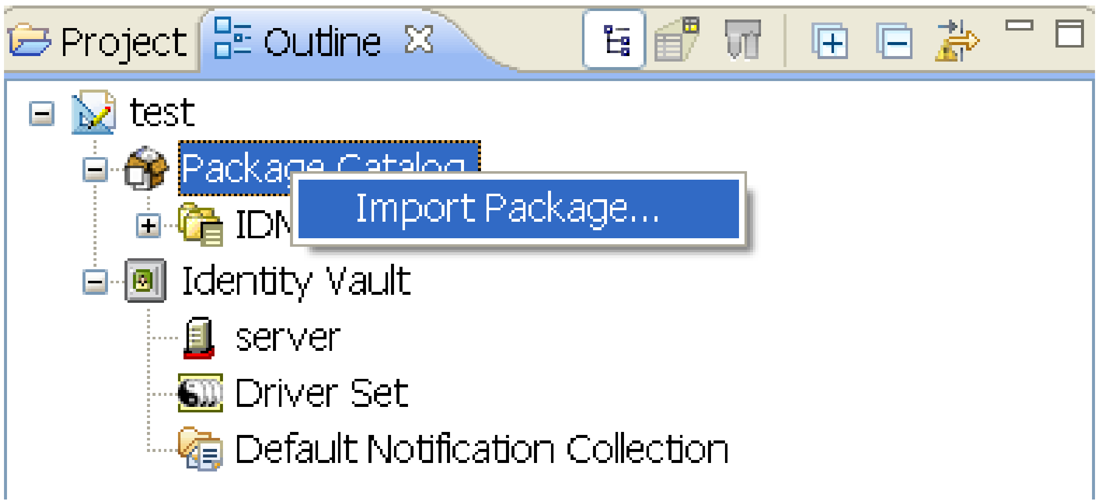
The driver packages contain the items required to create a driver, such as policies, entitlements, filters, and Schema Mapping policies. These packages are only available in Designer and can be updated after they are initially installed. You must have the most current version of the packages in the Package Catalog before you can create a new driver object.

To verify that you have the most recent version of the driver packages in the Package Catalog:

1. Open Designer.
2. In the toolbar, click *Help* > *Check for Package Updates*.
3. Click *OK* to update the packages

   or

   Click *OK* if the packages are up-to-date.
4. In the Outline view, right-click the Package Catalog.
5. Click *Import Package*.



6. Select any JMS driver packages

   or

   Click *Select All* to import all of the packages displayed.

   By default, only the base packages are displayed. Deselect *Show Base Packages Only* to display all packages.
7. Click *OK* to import the selected packages, then click *OK* in the successfully imported packages message.
8. After the current packages are imported, continue with Section 4.1.2, "Installing the Driver Packages," on page 28.

4.1.2 Installing the Driver Packages

After you have imported the current driver packages into the Package Catalog, you can install the driver packages to create a new driver.

1. In Designer, open your project.
2. In the Modeler, right-click the driver set where you want to create the driver, then click *New* > *Driver*.
3. Select *JMS Base*, then click *Next*.
4. Select the corresponding package for one of the supported JMS vendors. The options are:
   - JMS JBoss
   - JMS SonicMQ
   - JMS WebSphere
   - JMS TIBCO
   - Other

   You can select only one package at a time.
5. Click *Next*.
6. On the Driver Information page, specify a name for the driver, then click *Next*.
7. Fill in the following fields for Remote Loader information:

   Connect To Remote Loader: Select *Yes* or *No* to determine if the driver will use the Remote Loader. For more information, see the *Identity Manager 4.0.1 Remote Loader Guide*.

If you select *No*, skip to Step 8. If you select *Yes*, use the following information to complete the configuration of the Remote Loader:

Host Name: Specify the IP address or DNS name of the server where the Remote Loader is installed and running.

Port: Specify the port number for this driver. Each driver connects to the Remote Loader on a separate port. The default value is 8090.

Remote Loader Password: Specify a password to control access to the Remote Loader. It must be the same password that is specified as the Remote Loader password on the Remote Loader.

Driver Password: Specify a password for the driver to authenticate to the Metadirectory server. It must be the same password that is specified as the Driver Object Password on the Remote Loader.

8 Click *Next*.

9 If you selected *Other* in Step 4, skip to Step 10. Otherwise, fill in the field to specify the Broker URL for the JMS vendor you selected in Step 4. The URL usually consists of a protocol (http), an IP address (255.255.255.255), and a port number (8080). For example: jnp://172.17.2.16:1099.

10 Click *Next*.

11 Review the summary of tasks that will be completed to create the driver, then click *Finish*.

12 After you have installed the driver, you must change the configuration for your environment. Proceed to Section 4.1.3, "Configuring the Driver," on page 29.

4.1.3 Configuring the Driver

After installing the driver packages, the driver will start. However, the basic configuration probably does not meet the requirements for your environment. You should complete the following tasks to configure the driver:

- **Configure the driver parameters:** There are many settings that can help you customize and optimize the driver. The settings are divided into categories such as Driver Configuration, Engine Control Values, and Global Configuration Values (GCVs). Although it is important for you to understand all of the settings, your first priority should be to review the Driver Parameters located on the Driver Configuration page.
- **Configure the driver filter:** Modify the driver filter to include the object classes and attributes you want synchronized between the Identity Vault and the JMS vendor.
- **Configure policies:** Modify the policies on the Subscriber and Publisher channels. For information about using policies, see the *Policies in Designer 4.0.1* or *Policies in iManager for Identity Manager 4.0.1* guide.

After completing the configuration tasks, continue with the next section, Deploying the Driver.

4.1.4 Deploying the Driver

After a driver is created in Designer, it must be deployed into the Identity Vault.

1 In Designer, open your project.

2 In the Modeler, right-click the driver icon or the driver line, then select *Live > Deploy*.

3 If you are authenticated to the Identity Vault, skip to Step 5; otherwise, specify the following information:

Host: Specify the IP address or DNS name of the server hosting the Identity Vault.

Username: Specify the DN of the user object used to authenticate to the Identity Vault.

Password: Specify the user's password.

4. Click *OK*.
5. Read the deployment summary, then click *Deploy*.
6. Read the message, then click *OK*.
7. Click *Define Security Equivalence* to assign rights to the driver.

   The driver requires rights to objects within the Identity Vault and to the input and output directories on the server. The Admin user object is most often used to supply these rights. However, you might want to create a DriversUser (for example) and assign security equivalence to that user. Whatever rights that the driver needs to have on the server, the DriversUser object must have the same security rights.

   7a Click *Add*, then browse to and select the object with the correct rights.

   7b Click *OK* twice.
8. Click *Exclude Administrative Roles* to exclude users that should not be synchronized.

   You should exclude any administrative User objects (for example, Admin and DriversUser) from synchronization.

   8a Click *Add*, then browse to and select the user object you want to exclude.

   8b Click *OK*.

   8c Repeat Step 8a and Step 8b for each object you want to exclude.

   8d Click *OK*.
9. Click *OK*.

4.1.5 Starting the Driver

When a driver is created, it is stopped by default. To make the driver work, you must start the driver and cause events to occur. Identity Manager is an event-driven system, so after the driver is started, it won't do anything until an event occurs.

To start the driver:

1. In Designer, open your project.
2. In the Modeler, right-click the driver icon or the driver line, then select *Live* > *Start Driver*.

For information about management tasks with the driver, see Chapter 6, "Managing the Driver," on page 35.

4.2 Creating the Driver in iManager

Drivers are created with packages, and iManager does not support packages. In order to create or modify drivers, you must use Designer. See Section 4.1, "Creating the Driver in Designer," on page 27.

4.3 Activating the Driver

If you create the driver in a driver set where you have already activated a driver that comes with the Integration Module for Messaging, the driver inherits the activation. If you created the driver in a driver set that has not been activated, you must activate the driver, with the Integration Module for Messaging activation, within 90 days. Otherwise, the driver stops working.

For information on activation, refer to "Activating Novell Identity Manager Products" in the *Identity Manager 4.0.1 Integrated Installation Guide*.

5 Upgrading an Existing Driver

The following sections provide information to help you upgrade an existing driver to version 4.0.1:

5.1 Supported Upgrade Paths

You can upgrade from any Identity Manager 3.5.*x* version of the JMS driver. Upgrading a pre-3.5.*x* version of the driver directly to version 4.0.1 is not supported.

5.2 What's New in Version 4.0.1

Driver content is delivered in packages instead of through a driver configuration file.

5.3 Upgrade Procedure

The process for upgrading the JMS driver is the same as for other Identity Manager drivers. For detailed instructions, see "Upgrading Drivers to Packages" in the *Identity Manager 4.0.1 Upgrade and Migration Guide*.

6 Managing the Driver

As you work with the driver, there are a variety of management tasks you might need to perform, including the following:

- Starting, stopping, and restarting the driver
- Viewing driver version information
- Using Named Passwords to securely store passwords associated with the driver
- Monitoring the driver's health status
- Backing up the driver
- Inspecting the driver's cache files
- Viewing the driver's statistics
- Using the DirXML Command Line utility to perform management tasks through scripts
- Securing the driver and its information
- Synchronizing objects
- Migrating and resynchronizing data
- Activating the driver
- Upgrading an existing driver

Because these tasks, as well as several others, are common to all Identity Manager drivers, they are included in one reference, the *Identity Manager 4.0.1 Common Driver Administration Guide*.

7 Troubleshooting

Viewing driver processes is necessary to analyze unexpected behavior. To view the driver processing events, use DSTrace. You should only use it during testing and troubleshooting the driver. Running DSTrace while the drivers are in production increases the utilization on the Identity Manager server and can cause events to process very slowly.

For information about configuring the driver to use DSTrace, see “Viewing Identity Manager Processes” in the *Identity Manager 4.0.1 Common Driver Administration Guide*.

The driver supports the following six trace levels:

Level	Description
0	Minimal tracing such as JMS Driver version, Build Stamp, and XDS Library
1	Information on Connection
2	Information on Messages
3	Verbose information on the messages that are sent or received, and the GUIDs
4	Information on JNDI session, Context, and Connection
5	Information on the methods and its signatures

A Driver Properties

This section provides information about the Driver Configuration and Global Configuration Values properties for the JMS driver. These are the only unique properties for the JMS driver. All other driver properties (Named Password, Engine Control Values, Log Level, and so forth) are common to all drivers. Refer to "Driver Properties" in the *Identity Manager 4.0.1 Common Driver Administration Guide* for information about the common properties.

The information is presented from the viewpoint of iManager. If a field is different in Designer, it is marked with an icon.

- Section A.1, "Driver Configuration," on page 39
- Section A.2, "Global Configuration Values," on page 48

A.1 Driver Configuration

In iManager:

1 Click to display the Identity Manager Administration page.

2 Open the driver set that contains the driver whose properties you want to edit:

  2a In the *Administration* list, click *Identity Manager Overview*.

  2b If the driver set is not listed on the *Driver Sets* tab, use the *Search In* field to search for and display the driver set.

  2c Click the driver set to open the Driver Set Overview page.

3 Locate the driver icon, then click the upper right corner of the driver icon to display the *Actions* menu.

4 Click *Edit Properties* to display the driver's properties page.

In Designer:

1 Open a project in the Modeler, then right-click the driver line and click *Properties* > *Driver Configuration.*

The Driver Configuration options are divided into the following sections:

- Section A.1.1, "Driver Module," on page 40
- Section A.1.2, "Driver Object Password (iManager Only)," on page 40
- Section A.1.3, "Authentication," on page 40
- Section A.1.4, "Startup Option," on page 41
- Section A.1.5, "Driver Parameters," on page 41
- Section A.1.6, "ECMAScript," on page 47
- Section A.1.7, "Global Configurations," on page 47

A.1.1 Driver Module

The driver module changes the driver from running locally to running remotely or the reverse.

Java: Used to specify the name of the Java class that is instantiated for the shim component of the driver. This class can be located in the `classes` directory as a class file, or in the `lib` directory as a `.jar` file. If this option is selected, the driver is running locally.

The name of the Java class is:`com.novell.idm.driver.jms.JMSDriverShim`

Native: This option is not used with the driver.

Connect to Remote Loader: Used when the driver is connecting remotely to the connected system. Designer includes two suboptions:

- **Remote Loader Client Configuration for Documentation:** Includes information on the Remote Loader client configuration when Designer generates documentation for the JMS driver.
- **Driver Object Password:** Specifies a password for the Driver object. If you are using the Remote Loader, you must enter a password on this page. Otherwise, the remote driver does not run. The Remote Loader uses this password to authenticate itself to the remote driver shim.

A.1.2 Driver Object Password (iManager Only)

Driver Object Password: Use this option to set a password for the driver object. If you are using the Remote Loader, you must enter a password on this page or the remote driver does not run. This password is used by the Remote Loader to authenticate itself to the remote driver shim.

A.1.3 Authentication

The Authentication section stores the information required to authenticate to the connected system.

Authentication ID: Specify a user application ID. This ID is used to pass Identity Vault subscription information to the application.

Example: `Administrator`

Authentication Context: Specify the IP address or name of the server the application shim should communicate with.

Remote Loader Connection Parameters: Used only if the driver is connecting to the application through the remote loader. The parameter to enter is `hostname=xxx.xxx.xxx.xxx port=xxxx kmo=certificatename`, when the host name is the IP address of the application server running the Remote Loader server and the port is the port the remote loader is listening on. The default port for the Remote Loader is 8090.

The `kmo` entry is optional. It is only used when there is an SSL connection between the Remote Loader and the Metadirectory engine.

Example: `hostname=10.0.0.1 port=8090 kmo=IDMCertificate`

Application Password: Specify the password for the user object listed in the *Authentication ID* field.

Remote Loader Password: Used only if the driver is connecting to the application through the Remote Loader. The password is used to control access to the Remote Loader instance. It must be the same password specified during the configuration of the Remote Loader on the connected system.

Cache limit (KB): Specify the maximum event cache file size (in KB). If it is set to zero, the file size is unlimited. Click *Unlimited* to set the file size to unlimited in Designer

A.1.4 Startup Option

The Startup Option section allows you to set the driver state when the Identity Manager server is started.

Auto start: The driver starts every time the Identity Manager server is started.

Manual: The driver does not start when the Identity Manager server is started. The driver must be started through Designer or iManager.

Disabled: The driver has a cache file that stores all of the events. When the driver is set to Disabled, this file is deleted and no new events are stored in the file until the driver state is changed to Manual or Auto Start.

Do not automatically synchronize the driver: This option only applies if the driver is deployed and was previously disabled. If this is not selected, the driver re-synchronizes the next time it is started.

A.1.5 Driver Parameters

The Driver Parameters section lets you configure the driver-specific parameters. When you change driver parameters, you tune driver behavior to align with your network environment.

The parameters are presented by category:

- "Driver Options" on page 41
- "Subscriber Options" on page 43
- "Publisher Options" on page 45

Driver Options

Default JMS version: Specifies the API version this driver should use when communicating with message brokers. If you are uncertain, 1.0.2 is the more widely adopted standard.

This setting is global for all message brokers.

Broker ID: Specifies an identifier for this broker by which it is known in the Identity Manager namespace.The default value is WebSphere MQ 6. To use a value other than the default, you need to specify it.

Show connected-related options: Displays connection-related parameters, such as JNDI connection factory names and usernames or passwords. Select *show* to display the following options.

- **Username:** Specify the username to authenticate to the message broker.
- **Password:** Specify the password to authenticate to the message broker.

  After entering the password, you need to re-enter it for validation.
- **Show queue connection factory options:** Select *show* to display the queue connection factory options.
  - **JNDI name:** Specify the JNDI name of the connection factory used to create the connections to the queues.
- **Show topic connection factory options:** Select *show* to display topic connection factory options.
  - **JNDI name:** Specify the JNDI name of the connection factory used to create connections to the topics.
  - **Client ID:** Specify the client ID used to create the durable topic subscriptions.

NOTE: Changing this value after durable subscriptions have been defined is not recommended. If it is changed, the Publisher is unable to unsubscribe from existing topic subscriptions unless the client ID is set to the same value the subscriptions were created with.

Show standard JNDI context properties: Select *show* to display the standard JNDI context properties for this message broker. These properties are primarily used to specify the URL, username, and password used to connect to or authenticate with this broker.

- **INITAL_CONTEXT_FACTORY:** The name that uniquely identifies this JNDI context property.
- **Value:** Specify the name of the Java class used to create a JNDI context for this message broker.
- **PROVIDER_URL:** The name that uniquely identifies this JNDI context property.
- **Value:** Specify the URL of this message broker. A URL usually contains a protocol, an IP address, and a port number.
- **SECURITY_CREDENTIALS:** The name that uniquely identifies this JNDI context property.
- **Value:** Specify the password used to authenticate to this message broker.
- **SECURITY_PRINCIPAL:** The name that uniquely identifies this JNDI context property.
- **Value:** Specify the username used to authenticate to this message broker.
- **URL_PKG_PREFIXES:** The name that uniquely identifies this JNDI context property.
- **Value:** Specify the value of this JNDI context property.
- **Show remaining standard properties:** Select *show* to display the remaining, less commonly used standard JNDI context properties.
  - **APPLET:** The name that uniquely identifies this JNDI context property.
  - **Value:** Specify the name of the applet using used.
  - **AUTHORITATIVE:** The name that uniquely identifies this JNDI context property.
  - **Value:** The value of this JNDI context property.
  - **BATCHSIZE:** The name that uniquely identifies this JNDI context property.
  - **Value:** The value of this JNDI context property.
  - **DNS_URL:** The name that uniquely identifies this JNDI context property.
  - **Value:** The value of this JNDI context property.
  - **LANGUAGE:** The name that uniquely identifies this JNDI context property.
  - **Value:** The value of this JNDI context property.
  - **OBJECT_FACTORIES:** The name that uniquely identifies this JNDI context property.
  - **Value:** The value of this JNDI context property.
  - **REFERRAL:** The name that uniquely identifies this JNDI context property.
  - **Value:** The value of this JNDI context property.
  - **SECURITY_AUTHENTICATION:** The name that uniquely identifies this JNDI context property.
  - **Value:** The value of this JNDI context property.
  - **SECURITY_PROTOCOL:** The name that uniquely identifies this JNDI context property.
  - **Value:** The value of this JNDI context property.
  - **STATE_FACTORIES:** The name that uniquely identifies this JNDI context property.
  - **Value:** The value of this JNDI context property.

Show vendor-specific JNDI context properties: Select *show* to display the vendor-specific JNDI context properties.

- **Name:** The name that uniquely identifies this JNDI context property.
- **Value:** Specify the value of this JNDI context property.

Subscriber Options

Disable subscriber: Select *yes* to prevent this channel from sending messages to the JMS provider.

Show default message options: Select *show* to display the options that are global to all messages.

- **Default message expiration (milliseconds):** In milliseconds, specify how long messages should live after they reach the destination. This setting is global for all sent messages.
- **Default message priority:** Select the priority of the message. The options are:
    - 0 (normal)
    - 1 (normal)
    - 2 (normal)
    - 3 (normal)
    - 4 (normal, default)
    - 5 (expedited)
    - 6 (expedited)
    - 7 (expedited)
    - 8 (expedited)
    - 9 (expedited)

    Specifying expedited delivery can result in “out-of-order” message processing. This setting is global for all sent messages.
- **Default message type:** Select the default message type as text or bytes. This setting is global for all sent messages.
- **Show default destination options:** Select *show* to display the parameters that show the properties sent with the message.

    Message properties can be used to prevent message loopback or to pass application-specific information in messages. These properties are global for all sent messages.
    - **Name:** Message property names beginning with “JMS” must match those defined by the JMS specification or third-party providers.

        Property names fall into three general categories:
        - Standard JMS properties. They usually begin with JMS or JMSX.
        - Provider-specific properties. They usually begin with JMS_.
        - Application-specific. Anything else.
    - **Value:** The value of the message property.

Show default destination options: Select *show* to display the options global to all destinations.

- **Default destination type:** Select whether all destinations are queues (default) or topics. This setting is global for all destinations.
- **Default omit message envelope:** Select whether the JMS message envelope should be omitted from received messages. This setting is global to all destinations.

- **Default receive timeout (seconds):** Select how long a channel should wait to receive a response to a sent message. The default value is 10 seconds. Permitted values are no wait and 1-25. This setting is global to all destinations.
- **Default message filter:** Select how the destinations filter receives the messages. The options are:
  - Receive all messages
  - Receive messages from this instance
  - Receive messages from this channel
  - Receive messages from the other channel
  - Block messages from this instance (default)
  - Block messages from this channel
  - Block messages from the other channel
  - Specify a custom message selector
- **Default message selector:** If you select *specify a custom message selector*, specify a custom message selector to filter received messages. Message selectors are like SQL WHERE clauses, such as JMSCorrelationID LIKE '%01=whatever%'.

  The % wildcard character can be used to disregard content before or after the part of a header or property value you are interested in filtering on. When used in tandem with a message filter, the message selector is appended to the end of the filter by using an AND operator.

Destination unique id: Specify the identifier for this destination by which it is known in the Identity Manager namespace. This name is also the durable subscription name for topics. This value must be unique per channel (Subscriber/Publisher).

Show additional destination options: Select *show* to display additional options for this selected destination.

- **Destination JNDI name:** Specify the identifier for this destination that is known in the JNDI namespace. This might not be the name the destination is known by to the broker. This value does not need to be unique.
- **Destination type:** Select whether the destination type is inherited, a topic, or a queue.
- **Destination mode:** Select whether the destination is used to send or receive messages.
- **Message type:** Select whether messages are sent as a text or as bytes.
- **Show message properties:** Select *show* to display message properties sent with messages. Message properties can be used to prevent message loopback or pass provider or application-specific information along with messages.
  - **Name:** The message property names beginning with JMS must match those defined by the JMS specification or third-party providers. Property names fall into three general categories:
    - Standard JMS properties. They usually begin with JMS or JMSX.
    - Provider-specific properties. They begin with JMS_.
    - Application-specific. Anything else.
  - **Value:** Specify the value of the message property.

Destination unique id: Specify the identifier by which this destination is known in the Identity Manager namespace. This name is also the durable subscription name for topics. This value must be unique per channel (Subscriber/Publisher).

Show additional destination options: Select *show* to display additional options for this selected destination.

- **Destination JNDI name:** Specify the identifier by which this destination is known in the JNDI namespace. This might or might not be the name the destination is known by to the message broker.

  This value does not need to be unique.
- **Destination type:** Select whether the destination is inherited, a queue, or a topic.
- **Destination mode:** Select whether the destination is used to send or receive messages.
- **Omit message envelope:** Select whether the JMS message envelope is omitted from messages received by this destination.
- **Receive timeout (seconds):** Select how long a channel should wait to receive a response to a sent message. The default value is 10 seconds. Permitted values can range from 1-25.
- **Message filter:** Select how this destination filter receives messages. The options are:
  - Receive all messages
  - Receive messages from this instance
  - Receive messages from this channel
  - Receive messages from the other channel
  - Block messages from this instance (default)
  - Block messages from this channel
  - Block messages from the other channel
  - Specify a custom message selector
- **Message selector:** If you selected *specify a custom message selector*, specify a custom message selector to filter received messages. Message selectors are like SQL WHERE clauses, such as JMSCorrelationID = *whatever*. When used in tandem with a message filter, the message selector is appended to the end of the filter by using an AND operator.

Publisher Options

Disable publisher: Select *yes* to prevent this channel from sending messages to the JMS provider.

Heartbeat interval (minutes): Specifies how many minutes of inactivity should elapse before this channel sends a heartbeat document. In practice, more than the number of minutes specified can elapse. That is, this parameter defines a lower bound.

Show default message options: Select *show* to display options global to all messages.

- **Default message expiration (milliseconds):** Specify how long the messages live after they reach a destination. Specify the time duration in milliseconds. 0 means the message lives indefinitely. This setting is global for all sent messages.
- **Default message priority:** Select the priority of the message. The options are:
  - 0 (normal)
  - 1 (normal)
  - 2 (normal)
  - 3 (normal)
  - 4 (normal, default)
  - 5 (expedited)

    - 6 (expedited)
    - 7 (expedited)
    - 8 (expedited)
    - 9 (expedited)

  Specifying expedited delivery can result in “out-of-order” message processing. This setting is global for all sent messages.

- **Default message type:** Select whether the messages type is text or bytes. This setting is global for all sent messages.
- **Show default message properties:** Select *show* to display the parameter that specifies the properties sent with messages.

  Message properties can be used to prevent message loopback or pass application-specific information in messages. These properties are global for all sent messages.

    - **Name:** The message property names beginning with JMS must match those defined by the JMS specification or third-party providers. Property names fall into three general categories:
        - Standard JMS properties. They usually begin with JMS or JMSX.
        - Provider-specific properties. They begin with JMS_.
        - Application-specific. Anything else.
    - **Value:** Specify the value of the message property.

Show default session options: Select *show* to display options that are global to all sessions.

- **Default message acknowledgement threshold:** Specify how many messages are received by a monitored destination before an acknowledgement is sent to the broker.

Show default destination options: Select *show* to display options that are global to all destinations.

- **Default destination type:** Select whether the default destination type is a queue (default) or a topic.
- **Default omit message envelope:** Select whether the JMS message envelope is omitted from the received messages. This setting is global for all destinations.
- **Default receive timeout (seconds):** Select how long a channel waits to receive a response to a sent message. The default value is 10 seconds. The permitted values range from 1-25 seconds.
- **Default message filter:** Select how the destination’s filter receives the messages. The options are:
    - Receive all messages
    - Receive messages from this instance
    - Receive messages from this channel
    - Receive messages from the other channel
    - Block messages from this instance (default)
    - Block messages from this channel
    - Block messages from the other channel
    - Specify a custom message selector
- **Default message selector:** If you selected *specify a custom message selector*, specify a custom message selector to filter received messages. Message selectors are like SQL WHERE clauses, such as JMSCorrelationID LIKE '%01=whatever%'.

The % wildcard character is used to disregard content before or after the part of a header or property value you are interested in filtering on. When used in tandem with a message filter, the message selector is appended to the end of the filter by using an AND operator.

- **Default polling interval (milliseconds):** Specify how often the destinations are polled for new messages (in milliseconds).

Destination unique id: Specify the identifier by which this destination is known in the Identity Manager namespace. This name is also the durable subscription name for topics. This value must be unique per channel (Subscriber/Publisher).

Show additional destination options: Select *show* to display parameters for this selected destination.

- **Destination JNDI name:** Specify the identifier for this destination that is known in the JNDI namespace. This might not be the name the destination is known by to the broker. This value does not need to be unique.
- **Destination type:** Select whether the destination type is inherited, a topic, or a queue.
- **Destination mode:** Select whether the destination is used to send or receive messages.
- **Message type:** Select whether messages are sent as a text or as bytes.
- **Show message properties:** Select *show* to display message properties sent with messages. Message properties can be used to prevent message loopback or pass provider or application-specific information along with messages.
    - **Name:** The message property names beginning with JMS must match those defined by the JMS specification or third-party providers. Property names fall into three general categories:
        - Standard JMS properties. They usually begin with JMS or JMSX.
        - Provider-specific properties. They begin with JMS_.
        - Application-specific. Anything else.
    - **Value:** Specify the value of the message property.

A.1.6 ECMAScript

Displays an ordered list of ECMAScript resource files. The files contain extension functions for the driver that Identity Manager loads when the driver starts. You can add additional files, remove existing files, or change the order in which the files are executed.

A.1.7 Global Configurations

Displays an ordered list of Global Configuration objects. The objects contain extension GCV definitions for the driver that Identity Manager loads when the driver is started. You can add or remove the Global Configuration objects, and you can change the order in which the objects are executed.

A.2 Global Configuration Values

Global configuration values (GCVs) are values that can be used by the driver to control functionality. GCVs are defined on the driver or on the driver set. Driver set GCVs can be used by all drivers in the driver set. Driver GCVs can be used only by the driver on which they are defined.

The JMS driver includes several GCVs that are created from driver parameters. When you modify the driver parameters, the GCVs are updated; likewise, when you modify the GCVs, the driver parameters are updated. These GCVs are created so that the driver parameter information can be more easily used in the driver's policies.

You can also add your own GCVs if you discover you need additional ones as you implement policies in the driver.

To access the driver's GCVs in iManager:

1 Click to display the Identity Manager Administration page.

2 Open the driver set that contains the driver whose properties you want to edit.

 2a In the *Administration* list, click *Identity Manager Overview*.

 2b If the driver set is not listed on the *Driver Sets* tab, use the *Search In* field to search for and display the driver set.

 2c Click the driver set to open the Driver Set Overview page.

3 Locate the driver icon, click the upper right corner of the driver icon to display the *Actions* menu, then click *Edit Properties*.

 or

 To add a GCV to the driver set, click *Driver Set*, then click *Edit Driver Set properties*.

To access the driver's GCVs in Designer:

1 Open a project in the Modeler.

2 Right-click the driver icon or line, then select *Properties > Global Configuration Values.*

 or

 To add a GCV to the driver set, right-clickthe driver set icon , then click *Properties > GCVs*.

Destination unique ID: Specifies the identifier by which this destination is known in the Identity Manager namespace. This name is also the durable subscription name for topics.

This value must be unique per channel (Subscriber/Publisher).

B Trace Levels

The driver supports the following trace levels:

Level	Description
0	No debugging
1–2	Identity Manager messages.
3-4	Previous level plus JNDI details, JMS connection, JMS destination, JMS session details, and connection close status.
5	Previous level plus Method Level logging and JMS Message details like TextEncoding and Content type.

For information about setting driver trace levels, see “Viewing Identity Manager Processes” in the *Identity Manager 4.0.1 Common Driver Administration Guide*.