

## Driver for Java<sup>\*</sup> Messaging Service Implementation Guide

# Novell<sup>®</sup> Identity Manager

**3.6.1**

October 12, 2009

[www.novell.com](http://www.novell.com)



## Legal Notices

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

Any products or technical information provided under this Agreement may be subject to U.S. export controls and the trade laws of other countries. You agree to comply with all export control regulations and to obtain any required licenses or classification to export, re-export or import deliverables. You agree not to export or re-export to entities on the current U.S. export exclusion lists or to any embargoed or terrorist countries as specified in the U.S. export laws. You agree to not use deliverables for prohibited nuclear, missile, or chemical biological weaponry end uses. For more information on exporting Novell software, see the [Novell International Trade Services Web page \(http://www.novell.com/info/exports/\)](http://www.novell.com/info/exports/). Novell assumes no responsibility for your failure to obtain any necessary export approvals.

Copyright © 2004 - 2009 Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher.

Novell, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at [Novell Legal Patents \(http://www.novell.com/company/legal/patents/\)](http://www.novell.com/company/legal/patents/) and one or more additional patents or pending patent applications in the U.S. and in other countries.

Novell, Inc.  
404 Wyman Street, Suite 500  
Waltham, MA 02451  
U.S.A.  
[www.novell.com](http://www.novell.com)

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

## **Novell Trademarks**

For a list of Novell trademarks, see [Trademarks \(http://www.novell.com/company/legal/trademarks/tmlist.html\)](http://www.novell.com/company/legal/trademarks/tmlist.html).

## **Third-Party Materials**

All third-party trademarks are the property of their respective owners.



# Contents

<b>About This Guide</b>	<b>7</b>
<b>1 Overview</b>	<b>9</b>
1.1 Supported JMS Vendors and Versions	9
1.2 Key Terminology	9
1.3 JMS Messaging Models	10
1.3.1 Point-to-Point Messaging	10
1.3.2 Publish/Subscribe Messaging	11
1.4 JMS Messages	11
1.4.1 Message Structure	11
1.4.2 Message Types	12
1.5 How Subscriber and Publisher Channels Work	12
1.5.1 Subscriber Channel	13
1.5.2 Publisher Channel	13
1.6 Support for Standard Driver Features	14
1.6.1 Local Platforms	14
1.6.2 Remote Platforms	14
1.6.3 Entitlements	14
1.6.4 Password Synchronization Support	15
1.6.5 Information Synchronized	15
1.7 Additional Resources	15
<b>2 Installing Driver Files</b>	<b>17</b>
<b>3 Configuring Messaging Vendors</b>	<b>19</b>
3.1 Installing IBM WebSphere MQ on Win32	19
3.1.1 Placing Prerequisite Jar Files and Scripts	19
3.1.2 Creating a Default Queue Manager	20
3.1.3 Creating a Server-Connection Channel and Queues	21
3.1.4 Starting the Publish/Subscriber Broker	21
3.1.5 Installing System Queues Necessary for Publish/Subscribe	21
3.1.6 Creating a User Account	21
3.1.7 Setting Up JMS	22
3.2 Installing on JBossMQ	23
3.3 Installing on JBoss Messaging	24
3.4 Installing on SonicMQ	25
3.4.1 Locating Prerequisite Jar Files	25
3.4.2 Running Scripts to Configure the Messaging System	26
3.5 Installing on TIBCO EMS	26
3.5.1 Locating Prerequisite Client Jar Files	26
3.5.2 Running Scripts to Configure the Messaging System	27
<b>4 Creating a New Driver</b>	<b>29</b>
4.1 Creating the Driver in Designer	29
4.1.1 Importing the Driver Configuration File	29
4.1.2 Configuring the Driver	30
4.1.3 Deploying the Driver	30

4.1.4	Starting the Driver . . . . .	31
4.2	Creating the Driver in iManager . . . . .	31
4.2.1	Importing the Driver Configuration File . . . . .	32
4.2.2	Configuring the Driver . . . . .	34
4.2.3	Starting the Driver . . . . .	34
4.3	Activating the Driver . . . . .	34
<b>5</b>	<b>Upgrading an Existing Driver</b>	<b>35</b>
5.1	Supported Upgrade Paths . . . . .	35
5.2	What's New in Version 3.6.1 . . . . .	35
5.3	Upgrade Procedure . . . . .	35
<b>6</b>	<b>Managing the Driver</b>	<b>37</b>
<b>7</b>	<b>Troubleshooting</b>	<b>39</b>
<b>A</b>	<b>Driver Properties</b>	<b>41</b>
A.1	Driver Configuration . . . . .	41
A.1.1	Driver Module . . . . .	41
A.1.2	Driver Object Password (iManager Only) . . . . .	42
A.1.3	Authentication . . . . .	42
A.1.4	Startup Option . . . . .	43
A.1.5	Driver Parameters . . . . .	44
A.2	Global Configuration Values . . . . .	54
<b>B</b>	<b>Trace Levels</b>	<b>57</b>

# About This Guide

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

- ♦ [Chapter 1, “Overview,” on page 9](#)
- ♦ [Chapter 2, “Installing Driver Files,” on page 17](#)
- ♦ [Chapter 4, “Creating a New Driver,” on page 29](#)
- ♦ [Chapter 5, “Upgrading an Existing Driver,” on page 35](#)
- ♦ [Chapter 3, “Configuring Messaging Vendors,” on page 19](#)
- ♦ [Chapter 6, “Managing the Driver,” on page 37](#)
- ♦ [Chapter 7, “Troubleshooting,” on page 39](#)
- ♦ [Appendix A, “Driver Properties,” on page 41](#)
- ♦ [Appendix B, “Trace Levels,” on page 57](#)

## 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, or go to [www.novell.com/documentation/feedback.html](http://www.novell.com/documentation/feedback.html) and enter your comments there.

## Documentation Updates

For the most recent version of this guide, visit the [Identity Manager 3.6.1 Driver Documentation Web site \(http://www.novell.com/documentation/idm36drivers\)](http://www.novell.com/documentation/idm36drivers).

## Additional Documentation

For documentation on Identity Manager and other drivers, see the [Identity Manager 3.6.1 Documentation Web site \(http://www.novell.com/documentation/idm36\)](http://www.novell.com/documentation/idm36).

## Documentation Conventions

In Novell® documentation, a greater-than symbol (>) is used to separate actions within a step and items in a cross-reference path.

A trademark symbol (®, ™, etc.) denotes a Novell trademark. An asterisk (\*) denotes a third-party trademark.

When a single pathname can be written with a backslash for some platforms or a forward slash for other platforms, the pathname is presented with a backslash. Users of platforms that require a forward slash, such as Linux\* or UNIX\*, should use forward slashes as required by your software.





# Overview

# 1

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 messaging accessible. 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:

- ♦ [Section 1.1, “Supported JMS Vendors and Versions,” on page 9](#)
- ♦ [Section 1.2, “Key Terminology,” on page 9](#)
- ♦ [Section 1.3, “JMS Messaging Models,” on page 10](#)
- ♦ [Section 1.4, “JMS Messages,” on page 11](#)
- ♦ [Section 1.5, “How Subscriber and Publisher Channels Work,” on page 12](#)
- ♦ [Section 1.6, “Support for Standard Driver Features,” on page 14](#)
- ♦ [Section 1.7, “Additional Resources,” on page 15](#)

## 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
- ♦ 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 10](#)
- ♦ [Section 1.3.2, “Publish/Subscribe Messaging,” on page 11](#)

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

**Table 1-1** *Abstract Names vs. Messaging Model Names*

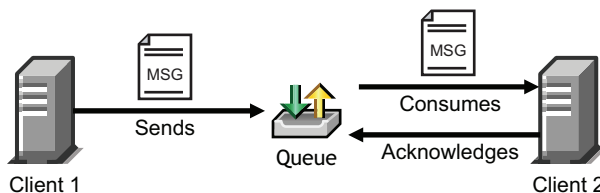
Abstract Terminology	Point-to-Point Terminology	Publish/Subscribe Terminology
Destination	Queue	Topic
Sender (or Producer)	Sender	Publisher
Receiver (or Consumer)	Receiver	Subscriber

### 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*

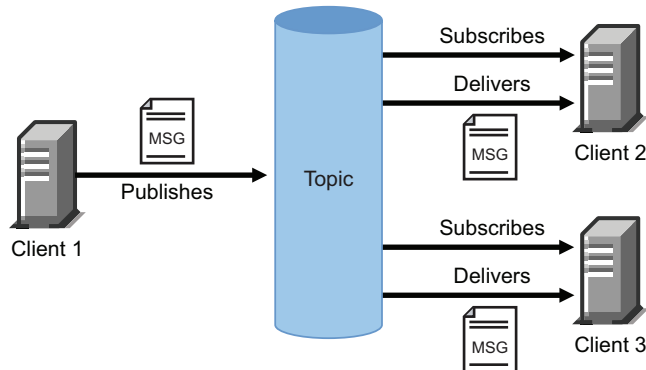


## 1.3.2 Publish/Subscribe Messaging

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 11](#)
- ♦ [Section 1.4.2, “Message Types,” on page 12](#)

### 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:headers>
</jms:message>
```

```

    <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 13](#)
- ♦ [Section 1.5.2, "Publisher Channel," on page 13](#)

## 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 name of the destinations used in these parameters can either be their 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 the first-defined send destination and does not wait for a message response (meaning that the message receipt is assumed to be asynchronous).

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 14](#)
- ◆ [Section 1.6.2, “Remote Platforms,” on page 14](#)
- ◆ [Section 1.6.3, “Entitlements,” on page 14](#)
- ◆ [Section 1.6.4, “Password Synchronization Support,” on page 15](#)
- ◆ [Section 1.6.5, “Information Synchronized,” on page 15](#)

### 1.6.1 Local Platforms

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 the Metadirectory Server](#)” in the *Identity Manager 3.6.1 Installation Guide*.

The JMS driver can be installed on the same operating systems supported by the Metadirectory server. For information about the operating systems supported by the Metadirectory server, see “[Metadirectory Server](#)” in “[System Requirements](#)” in the *Identity Manager 3.6.1 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 “[Remote Loader](#)” in the *Identity Manager 3.6.1 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 \(http://java.sun.com/products/jms/faq.html\)](http://java.sun.com/products/jms/faq.html)
- ♦ [Getting Started with JMS \(http://java.sun.com/developer/technicalArticles/Ecommerce/jms/index.html\)](http://java.sun.com/developer/technicalArticles/Ecommerce/jms/index.html)
- ♦ [JMS Tutorial \(http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html\)](http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html)
- ♦ [JMS Specifications \(1.0.2b and 1.1\) \(http://java.sun.com/products/jms/docs.html\)](http://java.sun.com/products/jms/docs.html)





# Installing Driver Files

# 2

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 29](#)) or upgrade an existing driver's configuration (see [Chapter 5, "Upgrading an Existing Driver," on page 35](#)).

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 the Metadirectory Server"](#) in the *Identity Manager 3.6.1 Installation Guide*.
- ♦ On a remote machine: Install the JMS driver files on the Remote Loader. See the instructions in ["Installing the Remote Loader"](#) in the *Identity Manager 3.6.1 Installation Guide*.



# Configuring Messaging Vendors

# 3

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

- ◆ [Section 3.1, “Installing IBM WebSphere MQ on Win32,” on page 19](#)
- ◆ [Section 3.2, “Installing on JBossMQ,” on page 23](#)
- ◆ [Section 3.3, “Installing on JBoss Messaging,” on page 24](#)
- ◆ [Section 3.4, “Installing on SonicMQ,” on page 25](#)
- ◆ [Section 3.5, “Installing on TIBCO EMS,” on page 26](#)

## 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.

- ◆ [Section 3.1.1, “Placing Prerequisite Jar Files and Scripts,” on page 19](#)
- ◆ [Section 3.1.2, “Creating a Default Queue Manager,” on page 20](#)
- ◆ [Section 3.1.3, “Creating a Server-Connection Channel and Queues,” on page 21](#)
- ◆ [Section 3.1.4, “Starting the Publish/Subscriber Broker,” on page 21](#)
- ◆ [Section 3.1.5, “Installing System Queues Necessary for Publish/Subscribe,” on page 21](#)
- ◆ [Section 3.1.6, “Creating a User Account,” on page 21](#)
- ◆ [Section 3.1.7, “Setting Up JMS,” on page 22](#)

### 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: 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 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\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 Default Queue Manager

The following instructions are for Windows; equivalent steps vary by platform.

- ♦ [“Manually Creating a Queue Manager through WebSphere MQ Explorer” on page 20](#)
- ♦ [“Setting Up Auto-Start for the Topic Message Broker” on page 20](#)

#### Manually Creating a Queue Manager through WebSphere MQ Explorer

- 1 Click *Start > Programs > IBM WebSphere MQ > WebSphere MQ Explorer*.
- 2 Right-click the *IBM WebSphere MQ > Queue Managers* folder.
- 3 Select *New > Queue Manager*.
- 4 Select the *Make this the default queue manager*.
- 5 Fill in the *Name* field.

If you don't name the queue, you might later see an error indicating “MQJMS5053:\*\*\* No broker response. Please ensure that the broker is running. If you are using the WebSphere MQ broker, check that your brokerVersion is set to V1.\*\*\*” This error is a side effect caused by the OS application event log filling up (on Windows).

- 6 Type `SYSTEM.DEAD.LETTER.QUEUE` for the *Dead Letter Queue* field.
- 7 Click the *Finish* button and wait for the operation to finish.
- 8 Continue with [“Setting Up Auto-Start for the Topic Message Broker” on page 20](#).

#### Setting Up Auto-Start for the Topic Message Broker

- 1 Click *Start > Programs > IBM WebSphere MQ > WebSphere MQ Explorer*

- 2 Right-click the *IBM WebSphere MQ > Queue Managers > Advanced > Services* folder.
- 3 Select *New > Service*.
- 4 Fill in the *Name* field. (For example, startBroker.)
- 5 Under *General*:
  - 5a Set the *Service control* field to *Queue Manager Start*.
  - 5b Set the *Start command* field to *strmqbrk*.
  - 5c Set the *Stop command* field to *endmqbrk*.
- 6 Click *Finish* and wait for a success message.

### 3.1.3 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.

### 3.1.4 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.

### 3.1.5 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."

---

### 3.1.6 Creating a User Account

- ♦ ["Creating a User" on page 21](#)
- ♦ ["Making the User a Member of the mqm Group" on page 22](#)

#### 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.

### 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 Left-click the mqm group
- 4 Click *Add*.
- 5 Click *OK* twice.

### 3.1.7 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://<hostname>:<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\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** Update the connection factory IP addresses and port in idm\_jms\_install.scp.

**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 may 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.

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 [“Show standard JNDI context parameters” on page 45](#)).

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.

### Generating a .bindings File

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.

## 3.2 Installing on JBossMQ

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.

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

- ♦ concurrent.jar

- ♦ jboss-common-client.jar
- ♦ jbossmq-client.jar
- ♦ jboss-system-client.jar
- ♦ jnp-system-client.jar
- ♦ jnp-client.jar
- ♦ log4j.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 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 [JBoss User Guide \(http://www.jboss.org/file-access/default/members/jbossmessaging/freezezone/docs/userguide-1.3.0.GA/html/index.html\)](http://www.jboss.org/file-access/default/members/jbossmessaging/freezezone/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: <code>novell\NDS\lib</code>
	Remote installation: <code>novell\RemoteLoader\lib</code>
Linux/UNIX	Local installation: <code>/usr/lib/dirxml/classes</code> (pre-eDirectory 8.8) or <code>/opt/novell/eDirectory/lib/dirxml/classes</code> (eDirectory 8.8)
	Remote installation: <code>/usr/lib/dirxml/classes</code> (pre-eDirectory 8.8) or <code>/opt/novell/eDirectory/lib/dirxml/classes</code> (eDirectory 8.8)

3 If necessary, restart your eDirectory server.

## 3.4 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 3.4.1, “Locating Prerequisite Jar Files,” on page 25](#)
- ♦ [Section 3.4.2, “Running Scripts to Configure the Messaging System,” on page 26](#)

### 3.4.1 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: <code>novell\NDS\lib</code> Remote installation: <code>novell\RemoteLoader\lib</code>
Linux/UNIX	Local installation: <code>/usr/lib/dirxml/classes</code> (pre-eDirectory 8.8) or <code>opt/novell/eDirectory/lib/dirxml/classes</code> (eDirectory 8.8) Remote installation: <code>/usr/lib/dirxml/classes</code> (pre-eDirectory 8.8) or <code>/opt/novell/eDirectory/lib/dirxml/classes</code> (eDirectory 8.8)

- 3 If necessary, restart your eDirectory server.

### 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.cli`) during the JMS driver installation. The following table indicates the default directories where scripts are installed, by platform.

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

- 2 Copy the script to your messaging server.
- 3 Follow the instructions provided in the script.

## 3.5 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 3.5.1, “Locating Prerequisite Client Jar Files,” on page 26](#)
- ♦ [Section 3.5.2, “Running Scripts to Configure the Messaging System,” on page 27](#)

### 3.5.1 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	<code>C:tibco\ems\clients\java</code>

Platform	Directory Path
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.

### 3.5.2 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\NDS\DirXMLUtilities\jms\tibco_ems
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
```

# 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 Driver Files,” on page 17](#)) and you’ve configured the JMS system (see [Chapter 3, “Configuring Messaging Vendors,” on page 19](#)), you can create the driver in the Identity Vault. You do so by importing the basic driver configuration file and then modifying the driver configuration to suit your environment. You can do this in either Designer or iManager.

- ♦ [Section 4.1, “Creating the Driver in Designer,” on page 29](#)
- ♦ [Section 4.2, “Creating the Driver in iManager,” on page 31](#)
- ♦ [Section 4.3, “Activating the Driver,” on page 34](#)

## 4.1 Creating the Driver in Designer

You create the JMS driver by importing the driver’s basic configuration file 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 Driver Configuration File,” on page 29](#)
- ♦ [Section 4.1.2, “Configuring the Driver,” on page 30](#)
- ♦ [Section 4.1.3, “Deploying the Driver,” on page 30](#)
- ♦ [Section 4.1.4, “Starting the Driver,” on page 31](#)

### 4.1.1 Importing the Driver Configuration File

There are four configuration files. Each file corresponds to one of the supported JMS vendors: JBossMQ, IBM WebSphereMQ, SonicMQ, and TIBCO EMS.

- 1 In Designer, open your project.
- 2 In the Modeler, right-click the driver set where you want to create the driver, then select *New > Driver* to display the Driver Configuration Wizard.
- 3 In the Driver Configuration list, select the desired JMS configuration, then click *Run*.
- 4 On the Import Information Requested page, fill in the following fields:

**Driver Name:** Specify a name that is unique within the driver set.

**JMS Broker Value:** Specify the URL for this message broker. The URL usually contains a protocol, an IP address, and a port number.

**Driver is Local/Remote:** Select *Local* if this driver will run on the Metadirectory server without using the Remote Loader service. Select *Remote* if you want the driver to use the Remote Loader service, either locally on the Metadirectory server or remotely on another server.

- 5 (Conditional) If you chose to run the driver remotely, click *Next*, then fill in the fields listed below. Otherwise, skip to [Step 6](#).

**Remote Host Name and Port:** Specify the host name or IP address of the server where the driver’s Remote Loader service is running.

**Driver Password:** Specify the driver object password that is defined in the Remote Loader service. The Remote Loader requires this password to authenticate to the Metadirectory server.

**Remote Password:** Specify the Remote Loader's password (as defined on the Remote Loader service). The Metadirectory engine (or Remote Loader shim) requires this password to authenticate to the Remote Loader

- 6 Click *Next* to import the driver configuration.

At this point, the driver is created from the basic configuration file. To ensure that the driver works the way you want it to for your environment, you must review and modify the driver's default configuration settings.

- 7 To review or modify the default configuration settings, click *Configure*, then continue with the next section, [Configuring the Driver](#).

or

To skip the configuration settings at this time, click *Close*. When you are ready to configure the settings, continue with [Configuring the Driver](#).

## 4.1.2 Configuring the Driver


After importing the driver configuration file, the driver will start. However, the basic configuration probably will 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 see the [Policies in Designer 3.5](#) or [Policies in iManager for Identity Manager 3.6.1](#) guide.

After completing the configuration tasks, continue with the next section, [Deploying the Driver](#).

## 4.1.3 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*.

#### 4.1.4 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 37](#).


## 4.2 Creating the Driver in iManager

You create the JMS driver by importing the driver's basic configuration file and then modifying the configuration to suit your environment. After you've created and configured the driver, you need to start it.

- ♦ [Section 4.2.1, "Importing the Driver Configuration File,"](#) on [page 32](#)
- ♦ [Section 4.2.2, "Configuring the Driver,"](#) on [page 34](#)
- ♦ [Section 4.2.3, "Starting the Driver,"](#) on [page 34](#)

## 4.2.1 Importing the Driver Configuration File

There are four configuration files. Each file corresponds to one of the supported JMS vendors: JBossMQ, IBM WebSphereMQ, SonicMQ, and TIBCO EMS.

- 1 In iManager, click  to display the Identity Manager Administration page.
- 2 In the Administration list, click *Import Configuration* to launch the Import Configuration wizard.
- 3 Follow the wizard prompts, filling in the requested information (described below) until you reach the Summary page.

Prompt	Description
Where do you want to place the new driver?	You can add the driver to an existing driver set, or you can create a new driver set and add the driver to the new set. If you choose to create a new driver set, you are prompted to specify the name, context, and server for the driver set.
Import a configuration into this driver set	Use the default option, <i>Import a configuration from the server (.XML file)</i> .  In the <i>Show</i> field, select <i>Identity Manager 3.6.1 configurations</i> .  In the <i>Configurations</i> field, select the JMS configuration file that corresponds to your JMS vendor.
Driver name	Type a name for the driver. The name must be unique within the driver set.
JMS Broker Value	Specify the URL for this message broker. The URL usually contains a protocol, an IP address, and a port number.
Driver is Local/Remote	Select <i>Local</i> if this driver will run on the Metadirectory server without using the Remote Loader service. Select <i>Remote</i> if you want the driver to use the Remote Loader service, either locally on the Metadirectory server or remotely on another server.
Remote Host Name and Port	This applies only if the driver is running remotely.  Specify the host name or IP address of the server where the driver's Remote Loader service is running.
Driver Password	This applies only if the driver is running remotely.  Specify the driver object password that is defined in the Remote Loader service. The Remote Loader requires this password to authenticate to the Metadirectory server.
Remote Password	This applies only if the driver is running remotely.  Specify the Remote Loader's password (as defined on the Remote Loader service). The Metadirectory engine (or Remote Loader shim) requires this password to authenticate to the Remote Loader



Prompt	Description
Define Security Equivalences	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.
Exclude Administrative Roles	You should exclude any administrative User objects (for example, Admin and DriversUser) from synchronization.

When you finish providing the information required by the wizard, a Summary page, similar to the following is displayed.

The following summarizes the state of the driver as it currently exists.

- [Warthog](#) (NCP Server)
- [DS2](#) (Driver Set)
- [JMS-JBOSS](#) (Drivers May Require Configuration) (Driver)
  - [smp-SchemaMap](#) (Schema Mapping Policy)
  - [pub-its-StripJMSMessageEnvelope](#) (Input Transformation Policy)
  - [sub-otp-SubAddAssociation](#) (Output Transformation Policy)
- [Publisher](#) (Publisher)
  - [none](#) (Command Transformation Policy)
  - [pub-ets-BlockLoopbackAdd](#) (Event Transformation Policy)
  - [none](#) (Matching Policy)
  - [none](#) (Creation Policy)
  - [pub-pp-MirrorPlacement](#) (Placement Policy)
- [Subscriber](#) (Subscriber)
  - [none](#) (Command Transformation Policy)

<< Back    Next >>    Cancel    Finish

At this point, the driver is created from the basic configuration file. To ensure that the driver works the way you want it to for your environment, you must review and modify the driver's default configuration settings.

- To modify the default configuration settings, click the linked driver name, then continue with the next section, [Configuring the Driver](#).

or

To skip the configuration settings at this time, click *Finish*. When you are ready to configure the settings, continue with [Configuring the Driver](#).

## 4.2.2 Configuring the Driver

After importing the driver configuration file, the driver will start. However, the basic configuration probably will 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 3.5](#) or [Policies in iManager for Identity Manager 3.6.1](#) guide.

After completing the configuration tasks, continue with the next section, [Starting the Driver](#).

## 4.2.3 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 iManager, click  to display the Identity Manager Administration page.
- 2 Click *Identity Manager Overview*.
- 3 Browse to and select the driver set object that contains the driver you want to start.
- 4 Click the driver set name to access the Driver Set Overview page.
- 5 Click the upper right corner of the driver, then click *Start driver*.

For information about management tasks with the driver, see [Chapter 6, “Managing the Driver,”](#) on [page 37](#).

## 4.3 Activating the Driver

If you create the driver in a driver set where you've 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 3.6.1 Installation Guide](#).

# Upgrading an Existing Driver

# 5

The following sections provide information to help you upgrade an existing driver to version 3.6.1:

- ♦ [Section 5.1, “Supported Upgrade Paths,” on page 35](#)
- ♦ [Section 5.2, “What’s New in Version 3.6.1,” on page 35](#)
- ♦ [Section 5.3, “Upgrade Procedure,” on page 35](#)

## 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 3.6.1 is not supported.

## 5.2 What’s New in Version 3.6.1

- ♦ Support for Tibco EMS 5.0 has been added.
- ♦ Support for JBoss messaging 1.3.0 has been added.

## 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](#)” in the *Identity Manager 3.6.1 Installation Guide*.



# Managing the Driver

# 6

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<sup>®</sup> 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 3.6.1 Common Driver Administration Guide*.



# Troubleshooting

# 7

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 more information about the trace levels supported by the JMS driver, see [Appendix B, “Trace Levels,”](#) on page 57.

For information about configuring the driver to use DSTrace, see “[Viewing Identity Manager Processes](#)” in the *Identity Manager 3.6.1 Common Driver Administration Guide*.






# Driver Properties

# A


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 3.6.1 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 41](#)
- ♦ [Section A.2, “Global Configuration Values,” on page 54](#)

## 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 41](#)
- ♦ [Section A.1.2, “Driver Object Password \(iManager Only\),” on page 42](#)
- ♦ [Section A.1.3, “Authentication,” on page 42](#)
- ♦ [Section A.1.4, “Startup Option,” on page 43](#)
- ♦ [Section A.1.5, “Driver Parameters,” on page 44](#)

### A.1.1 Driver Module

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

**Table A-1** *Driver Modules*

Option	Description
<i>Java</i>	<p>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 <code>classes</code> directory as a class file, or in the <code>lib</code> directory as a <code>.jar</code> file. If this option is selected, the driver is running locally.</p> <p>The name of the Java class is:</p> <pre>com.novell.idm.driver.jms.JMSDriverShim</pre>
<i>Native</i>	<p>This option is not used with the driver.</p>
<i>Connect to Remote Loader</i>	<p>Used when the driver is connecting remotely to the connected system. Designer includes two suboptions:</p> <ul style="list-style-type: none"><li>◆  <b>Driver Object Password:</b> 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.</li><li>◆  <b>Remote Loader Client Configuration for Documentation:</b> Includes information on the Remote Loader client configuration when Designer generates documentation for the Delimited Text driver.</li></ul>

## A.1.2 Driver Object Password (iManager Only)


**Table A-2** *Driver Object Password*










Option	Description
<i>Driver Object Password</i>	<p>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.</p>

## A.1.3 Authentication

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

**Table A-3** *Authentication*

Option	Description
<i>Authentication ID</i> or  <i>User ID</i>	<p>Specify a user application ID. This ID is used to pass Identity Vault subscription information to the application.</p> <p>Example: Administrator</p>


Option	Description
<p><i>Authentication Context</i></p> <p>or</p> <p> <i>Connection Information</i></p>	Specify the IP address or name of the server the application shim should communicate with.
<p><i>Remote Loader Connection Parameters</i></p> <p>or</p> <p> <i>Host name</i></p> <p> <i>Port</i></p> <p> <i>KMO</i></p> <p> <i>Other parameters</i></p>	<p>Used only if the driver is connecting to the application through the remote loader. The parameter to enter is</p> <pre>hostname=xxx.xxx.xxx.xxx port=xxxx</pre> <p><i>kmo=certificatename</i>, 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.</p> <p>The <i>kmo</i> entry is optional. It is only used when there is an SSL connection between the Remote Loader and the Metadirectory engine.</p> <p>Example: <code>hostname=10.0.0.1 port=8090</code>  <code>kmo=IDMCertificate</code></p>
<p><i>Driver Cache Limit (kilobytes)</i></p> <p>or</p> <p> <i>Cache limit (KB)</i></p>	<p>Specify the maximum event cache file size (in KB). If it is set to zero, the file size is unlimited.</p> <p> Click <i>Unlimited</i> to set the file size to unlimited in Designer.</p>
<p><i>Application Password</i></p> <p>or</p> <p> <i>Set Password</i></p>	Specify the password for the user object listed in the <i>Authentication ID</i> field.
<p><i>Remote Loader Password</i></p> <p>or</p> <p> <i>Set Password</i></p>	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.

## A.1.4 Startup Option

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

**Table A-4** *Startup Option*

Option	Description
<i>Auto start</i>	The driver starts every time the Identity Manager server is started.
<i>Manual</i>	The driver does not start when the Identity Manager server is started. The driver must be started through Designer or iManager.
<i>Disabled</i>	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.

Option	Description
 <i>Do not automatically synchronize the driver</i>	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:

- ◆ [Table A-5, “Driver Settings,” on page 44](#)
- ◆ [Table A-6, “Subscriber Settings,” on page 46](#)
- ◆ [Table A-7, “Publisher Settings,” on page 51](#)

**Table A-5** *Driver Settings*

Option	Sub-Option	Description
<i>Default JMS version</i>		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.
<i>Broker ID</i>		Specifies an identifier for this broker by which it is known in the Identity Manager namespace.
<i>Show connection-related parameters</i>		Displays connection-related parameters such as JNDI connection factory names and usernames or passwords.
	<i>Username</i>	Specifies the username used to authenticate to the message broker.
	<i>Password</i>	The password used to authenticate to the message broker.  After entering the password, you need to re-enter it for validation.
	<i>Show queue connection factory options</i>	Specifies the JNDI name of the connection factory used to create connections to queues.

Option	Sub-Option	Description
	<i>Show topic connection factory options</i>	<ol style="list-style-type: none"> <li>1. Specify the JNDI name of the connection factory used to create connections to topics.</li> <li>2. Specify the Client ID used to create durable topic subscriptions.</li> </ol> <hr/> <p><b>NOTE:</b> 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.</p> <hr/>
<i>Show standard JNDI context parameters</i>		Displays 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.
	<i>INITIAL_CONTEXT_FACTORY</i>	The name that uniquely identifies this JNDI context property.
	<i>Value</i>	The name of the Java class used to create a JNDI context for this message broker.
prov	<i>PROVIDER_URL</i>	The name that uniquely identifies this JNDI context property
	<i>Value</i>	The URL of this message broker. A URL usually contains a protocol, an IP address, and a port number.  For example, jnp://140.67.155.9:1099
	<i>SECURITY_CREDENTIALS</i>	The name that uniquely identifies this JNDI context property.
	<i>Value</i>	The password used to authenticate to this message broker.
	<i>SECURITY_PRINCIPAL</i>	The name that uniquely identifies this JNDI context property.
	<i>Value</i>	The username used to authenticate to this message broker.
	<i>URL_PKG_PREFIXES</i>	The name that uniquely identifies this JNDI context property.
	<i>Value</i>	The value of this JNDI context property.
	<i>Show remaining standard properties</i>	Displays remaining, less commonly used standard JNDI context properties
	<i>APPLET</i>	The name that uniquely identifies this JNDI context property.
	<i>Value</i>	The value of this JNDI context property.

Option	Sub-Option	Description
	<i>AUTHORITATIVE</i>	The name that uniquely identifies this JNDI context property.
	<i>Value</i>	The value of this JNDI context property.
	<i>BATCHSIZE</i>	The name that uniquely identifies this JNDI context property.
	<i>Value</i>	The value of this JNDI context property.
	<i>DNS_URL</i>	The name that uniquely identifies this JNDI context property.
	<i>Value</i>	The value of this JNDI context property.
	<i>LANGUAGE</i>	The name that uniquely identifies this JNDI context property.
	<i>Value</i>	The value of this JNDI context property.
	<i>OBJECT_FACTORIES</i>	The name that uniquely identifies this JNDI context property.
	<i>Value</i>	The value of this JNDI context property.
	<i>REFERRAL</i>	The name that uniquely identifies this JNDI context property.
	<i>Value</i>	The value of this JNDI context property.
	<i>SECURITY_AUTHENTICATION</i>	The name that uniquely identifies this JNDI context property.
	<i>Value</i>	The value of this JNDI context property.
	<i>SECURITY_PROTOCOL</i>	The name that uniquely identifies this JNDI context property.
	<i>Value</i>	The value of this JNDI context property.
	<i>STATE_FACTORIES</i>	The name that uniquely identifies this JNDI context property.
	<i>Value</i>	The value of this JNDI context property.
<i>Show vendor-specific JNDI context properties</i>		Displays vendor-specific JNDI context properties.
	<i>Name</i>	The name that uniquely identifies this JNDI context property.
	<i>Value</i>	The value of this JNDI context property.

**Table A-6** *Subscriber Settings*

Option	Sub-Option	Description
<i>Disable subscriber</i>		Select <i>yes</i> to prevent this channel from sending messages to JMS providers.

Option	Sub-Option	Description
<i>Show default message options</i>		Displays options that are global to all messages.
	<i>Default message expiration (milliseconds)</i>	How long messages should live after they reach a destination. Specify the time duration in milliseconds. This setting is global for all sent messages.  A value of 0 specifies that the message lives indefinitely.
	<i>Default message priority</i>	Specifies the message priority.  0-4 indicates normal delivery 5-9 indicates expedited delivery  Specifying expedited delivery can result in “out-of-order” message processing. This setting is global for all sent messages.
	<i>Default message type</i>	Specifies the default message type as text or bytes. This setting is global for all sent messages.
	<i>Show default message properties</i>	Specifies whether to display parameters that show the properties sent with messages.  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.
	<i>Name</i>	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:  1. Standard JMS properties. They usually begin with JMS or JMSX. 2. Provider-specific properties. They usually begin with JMS_. 3. Application-specific. Anything else.
<i>Show default destination options</i>	<i>Value</i>	Message property value.
		Displays options global to all destinations.
	<i>Default destination type</i>	Specifies whether all destinations are queues or topics by default. This setting is global for all destinations.
	<i>Default omit message envelope</i>	Specifies whether the JMS message envelope should be omitted from received messages. This setting is global to all destinations.

Option	Sub-Option	Description
	<i>Default receive timeout (seconds)</i>	<p>Specifies 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.</p> <p>This setting is global to all destinations.</p>
	<i>Default message filter</i>	<p>Specifies how destinations filter received messages.</p> <p>This setting is global to all destinations.</p>
	<i>Default message selector</i>	<p>Specifies a custom message selector to filter received messages. Message selectors are like SQL WHERE clauses, such as JMSCorrelationID LIKE '%01=whatever%'.</p> <p>The % wildcard character can be used to disregard content before or after the part of a header or property value you're 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.</p>
<i>Destination unique id</i>		<p>Specifies 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).</p>
<i>Show additional destination options</i>		<p>Displays additional options for this selected destination.</p>
	<i>Destination JNDI name</i>	<p>Specifies 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.</p>
	<i>Destination type</i>	<p>Specifies whether this destination is a queue or a topic.</p>
	<i>Destination mode</i>	<p>Specifies whether the destination is used to send or receive messages.</p>
	<i>Message type</i>	<p>Specifies whether messages are sent as text or bytes.</p>
	<i>Show message properties</i>	<p>Specifies whether 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.</p>



Option	Sub-Option	Description
	<i>Name</i>	<p>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:</p> <ol style="list-style-type: none"> <li>1. Standard JMS properties. They usually begin with JMS or JMSX.</li> <li>2. Provider-specific properties. They begin with JMS_.</li> <li>3. Application-specific. Anything else.</li> </ol>
	<i>Value</i>	Message property value.
<i>Destination unique id</i>		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).
<i>Show additional destination options</i>		Displays additional options for this selected destination.
	<i>Destination JNDI name</i>	<p>Specifies 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.</p> <p>This value does not need to be unique.</p>
	<i>Destination type</i>	Specifies whether the destination is a queue or a topic.
	<i>Destination mode</i>	Specifies whether the destination is used to send or receive messages.
	<i>Message type</i>	Specifies whether messages should be sent as text or bytes.
	<i>Show message properties</i>	Displays options that specify the message properties sent with messages. Message properties can be used to prevent message loopback or pass provider or application-specific information along with messages.
	<i>Name</i>	<p>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:</p> <ol style="list-style-type: none"> <li>1. Standard JMS properties. They usually begin with JMS or JMSX.</li> <li>2. Provider-specific properties. They begin with JMS_.</li> <li>3. Application-specific. Anything else.</li> </ol>
	<i>Value</i>	Message property value.

Option	Sub-Option	Description
	<i>Omit message envelope</i>	Specifies whether the JMS message envelope is omitted from messages received by this destination.
	<i>Receive timeout (seconds)</i>	Specifies 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.
	<i>Message filter</i>	Specifies how the destination receives filtered messages.
	<i>Message selector</i>	Specifies a custom message selector to filter received messages. Message selectors are like SQL WHERE clauses (for example, JMSCorrelationID = <i>whatever</i> ). When used in tandem with a message filter, the message selector is appended to the end of the filter by using an AND operator.
<i>Destination unique id</i>		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).
<i>Show additional destination options</i>		Displays additional options for this destination.
	<i>Destination JNDI name</i>	Specifies 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.
	<i>Destination type</i>	Specifies whether the destination is a queue or a topic.
	<i>Destination mode</i>	Specifies whether the destination is used to send or receive messages.
	<i>Message type</i>	Specifies whether messages should be sent as text or bytes.
	<i>Show message properties</i>	Displays options that specify the message properties sent with messages. Message properties can be used to prevent message loopback or pass provider/application-specific information along with messages.

Option	Sub-Option	Description
	<i>Name</i>	<p>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:</p> <ol style="list-style-type: none"> <li>1. Standard JMS properties. They usually begin with JMS or JMSX.</li> <li>2. Provider-specific properties. They begin with JMS_.</li> <li>3. Application-specific. Anything else.</li> </ol>
	<i>Value</i>	Message property value.

**Table A-7** *Publisher Settings*

Option	Sub-Option	Description
<i>Disable publisher</i>		Select yes to prevent this channel from receiving messages from JMS providers.
<i>Heartbeat interval (minutes)</i>		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.
<i>Show default message options</i>		Displays options global to all messages.
	<i>Default message expiration (milliseconds)</i>	<p>Specifies how long messages live after they reach a destination.</p> <p>Specify the time duration in milliseconds. 0 means the messages live indefinitely. This setting is global for all sent messages.</p>
	<i>Default message priority</i>	<p>Specifies the message priority.</p> <p>0-4 indicates normal delivery 5-9 indicates expedited delivery</p> <p>Specifying expedited delivery can result in “out-of-order” message processing. This setting is global for all sent messages.</p>
	<i>Default message type</i>	Specifies whether messages are text or byte. This setting is global for all sent messages.
	<i>Show default message properties</i>	<p>Displays the parameters that specify the properties sent with messages.</p> <p>Message properties can be used to prevent message loopback or pass application-specific information in messages.</p> <p>These properties are global for all sent messages.</p>

Option	Sub-Option	Description
<i>Show default session options</i>	<i>Name</i>	<p>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:</p> <ol style="list-style-type: none"> <li>1. Standard JMS properties. They usually begin with JMS or JMSX.</li> <li>2. Provider-specific properties. They begin with JMS_.</li> <li>3. Application specific. Anything else.</li> </ol>
	<i>Value</i>	<p>Message property value.</p> <p>Displays options that are global to all sessions.</p>
<i>Show default destination options</i>	<i>Default message acknowledgment threshold</i>	<p>Specifies how many messages are received by a monitored destination before an acknowledgment is sent to the broker.</p>
	<i>Default destination type</i>	<p>Displays options that are global to all destinations.</p> <p>Specifies whether destinations are topics or queues by default.</p> <p>This setting is global for all destinations.</p>
	<i>Default omit message envelope</i>	<p>Specifies if the JMS message envelope is omitted from received messages.</p> <p>This setting is global for all destinations.</p>
	<i>Default receive timeout (seconds)</i>	<p>Specifies how long a channel waits to receive a response to a sent message. The default is 10 seconds. Permitted values range from 1-25 seconds.</p> <p>This setting is global for all destinations.</p>
	<i>Default message filter</i>	<p>Specifies how the destination's filter receives messages.</p> <p>This setting is global for all destinations.</p>
	<i>Default message selector</i>	<p>Specifies a custom message selector to filter received messages. Message selectors are like SQL WHERE clauses, such as, JMSCorrelationID LIKE '%01=whatever%'.</p> <p>The % wildcard character is used to disregard content before or after the part of a header or property value you're 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.</p>

Option	Sub-Option	Description
	<i>Default polling interval (milliseconds)</i>	Specifies how often destinations are polled for new messages (in milliseconds.)  This setting is global for all destinations.
<i>Destination unique id</i>		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).
<i>Show additional destination options</i>		Displays parameters for this selected destination.
	<i>Destination JNDI name</i>	Specifies the identifier by which this destination 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.
	<i>Destination type</i>	Specifies whether this destination is a queue or topic.
	<i>Destination mode</i>	Specifies whether this destination sends or receives messages.
	<i>Message type</i>	Specifies whether messages are sent in text or byte format.
	<i>Show message properties</i>	Displays the parameters that specify the properties sent with messages.  Message properties can be used to prevent message loopback or pass application-specific information along with messages.
	<i>Name</i>	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: <ol style="list-style-type: none"> <li>1. Standard JMS properties. They usually begin with JMS or JMSX.</li> <li>2. Provider-specific properties. They begin with JMS_.</li> <li>3. Application-specific. Anything else.</li> </ol>
	<i>Value</i>	Message property value.
<i>Destination unique id</i>		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).
<i>Show additional destination options</i>		Displays parameters for this selected destination.

Option	Sub-Option	Description
	<i>Destination JNDI name</i>	Specifies 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 broker. This value does not need to be unique.
	<i>Destination type</i>	Specifies whether the destination is a queue or a topic.
	<i>Destination mode</i>	Specifies whether this destination sends or receives messages.
	<i>Omit message envelope</i>	Whether the JMS message envelope be omitted from messages received by this destination.
	<i>Receive timeout (seconds)</i>	Specifies how long this channel waits to receive a response from a destination. The default is 10 seconds. Permitted values range from 1-25 seconds.
	<i>Message filter</i>	Specifies how this destination filters messages.
	<i>Message selector</i>	Specifies a custom message selector to filter received messages. Message selectors are like SQL WHERE clauses For example, <code>JMSCorrelationID = whatever</code> . When used in tandem with a message filter, the message selector is appended to the end of the filter by using an AND operator.
	<i>Is durable</i>	Specifies whether messages are cached at the message broker when the driver isn't running.  This setting is only effective for topic destinations; queues are durable by default.
	<i>Subscription name</i>	Specify the name of the durable subscription to create on the broker.  <b>NOTE:</b> This resource might need to be cleaned up manually when this driver is deleted unless specific procedures are followed.
	<i>Actively monitor</i>	Specify if you want the channel to periodically monitor this destination for messages.


## 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-click the driver set icon , then click *Properties > GCVs*.

**Table A-8** *Global Configuration Values*

Option	Description
<i>Destination unique ID</i>	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).





# Trace Levels

# B

The driver supports the following six trace levels:

**Table B-1** *Supported 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

For information about setting driver trace levels, see to “[Viewing Identity Manager Processes](#)” in the *Identity Manager 3.6.1 Common Driver Administration Guide*.

