NetIQ® Identity Manager
Driver for Java Messaging Service
Implementation Guide

February 2018

NetIQ.

Legal Notices

For information about NetIQ trademarks, see https://www.netiq.com/company/legal/.

Contents

About this Book and the Library

The *Identity Manager Driver for JMS Implementation Guide* explains how to install and configure the Identity Manager Driver for Java Messaging Service (JMS).

Intended Audience

This book provides information for individuals responsible for understanding administration concepts and implementing a secure, distributed administration model.

Other Information in the Library

For more information about the library for Identity Manager, see the following resources:

- Identity Manager documentation website (https://www.netiq.com/documentation/identity-manager-47/)
- Identity Manager drivers documentation website (https://www.netiq.com/documentation/identity-manager-47-drivers/)

About NetIQ Corporation

We are a global, enterprise software company, with a focus on the three persistent challenges in your environment: Change, complexity, and risk—and how we can help you control them.

Our Viewpoint

Adapting to change and managing complexity and risk are nothing new

In fact, of all the challenges you face, these are perhaps the most prominent variables that deny you the control you need to securely measure, monitor, and manage your physical, virtual, and cloud computing environments.

Enabling critical business services, better and faster

We believe that providing as much control as possible to IT organizations is the only way to enable timelier and cost effective delivery of services. Persistent pressures like change and complexity will only continue to increase as organizations continue to change and the technologies needed to manage them become inherently more complex.

Our Philosophy

Selling intelligent solutions, not just software

In order to provide reliable control, we first make sure we understand the real-world scenarios in which IT organizations like yours operate—day in and day out. That's the only way we can develop practical, intelligent IT solutions that successfully yield proven, measurable results. And that's so much more rewarding than simply selling software.

Driving your success is our passion

We place your success at the heart of how we do business. From product inception to deployment, we understand that you need IT solutions that work well and integrate seamlessly with your existing investments; you need ongoing support and training post-deployment; and you need someone that is truly easy to work with—for a change. Ultimately, when you succeed, we all succeed.

Our Solutions

- Identity & Access Governance
- Access Management
- Security Management
- Systems & Application Management
- Workload Management
- Service Management

Contacting Sales Support

For questions about products, pricing, and capabilities, contact your local partner. If you cannot contact your partner, contact our Sales Support team.

Worldwide:	www.netiq.com/about_netiq/officelocations.asp
United States and Canada:	1-888-323-6768
Email:	info@netiq.com
Web Site:	www.netiq.com

Contacting Technical Support

For specific product issues, contact our Technical Support team.

Worldwide:	www.netiq.com/support/contactinfo.asp
North and South America:	1-713-418-5555
Europe, Middle East, and Africa:	+353 (0) 91-782 677
Email:	support@netiq.com
Web Site:	www.netiq.com/support

Contacting Documentation Support

Our goal is to provide documentation that meets your needs. The documentation for this product is available on the NetIQ Web site in HTML and PDF formats on a page that does not require you to log in. If you have suggestions for documentation improvements, click **Add Comment** at the bottom of any page in the HTML version of the documentation posted at www.netiq.com/documentation. You can also email Documentation-Feedback@netiq.com. We value your input and look forward to hearing from you.

Contacting the Online User Community

NetIQ Communities, the NetIQ online community, is a collaborative network connecting you to your peers and NetIQ experts. By providing more immediate information, useful links to helpful resources, and access to NetIQ experts, NetIQ Communities helps ensure you are mastering the knowledge you need to realize the full potential of IT investments upon which you rely. For more information, visit community.netiq.com.

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:

- "Supported JMS Vendors and Versions" on page 9
- "Key Terminology" on page 9
- "JMS Messaging Models" on page 10
- "JMS Messages" on page 11
- "How Subscriber and Publisher Channels Work" on page 13
- "Support for Standard Driver Features" on page 14
- "Additional Resources" on page 15

Supported JMS Vendors and Versions

The driver supports the following vendors and versions:

- JBossMQ v4.2.2
- JBoss Messaging 1.3.0, JBoss Messaging 1.4.6, and Messaging 2.4.0
- IBM WebSphere MQ v8.x and MQ v9.x
- SonicMQ v7.x, and Sonic 8.5
- TIBCO EMS v4, v5, and v8.0

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

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.)

JMS Messaging Models

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

- "Point-to-Point Messaging" on page 10
- "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.

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

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*
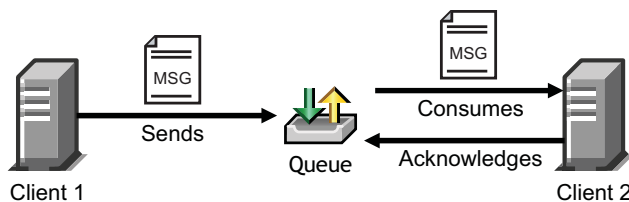
Publish/Subscribe Messaging

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*
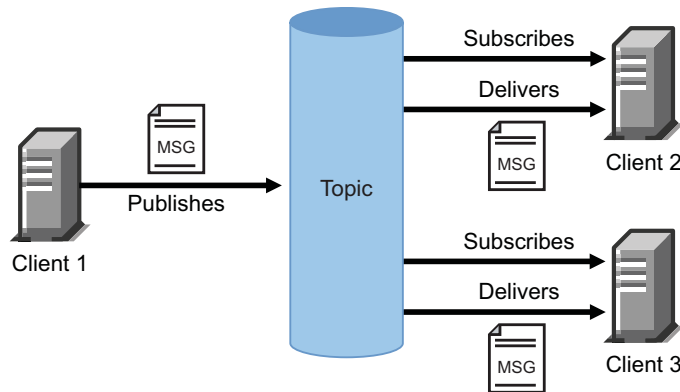
JMS Messages

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

- "Message Structure" on page 11
- "Message Types" on page 12

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 Identity Manager engine are also wrapped in an envelope. All message envelope elements and special attributes must have a namespace prefix bound to **urn:idm:jms**. If the namespace is missing on the Publisher channel, the JMS driver appends it by adding (**xmlns:jms="urn:idm:jms"**) in the incoming message. If a namespace exists, the driver overrides **"jms"** namespace with **"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>
```

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>
```

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.

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>
```

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.

Support for Standard Driver Features

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

- ◆ "Local Platforms" on page 14
- ◆ "Remote Platforms" on page 14
- ◆ "Entitlements" on page 15
- ◆ "Password Synchronization Support" on page 15
- ◆ "Information Synchronized" on page 15

Local Platforms

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

The JMS driver can be installed on the same operating systems that are supported by the Identity Manager server. For information about the operating systems supported by the Identity Manager server, see "Implementation Checklist" in the *NetIQ Identity Manager Setup Guide for Linux* or "Implementation Checklist" in the *NetIQ Identity Manager Setup Guide for Windows*.

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 Configuring the Remote Loader and Drivers in the *NetIQ Identity Manager Setup Guide for Linux* or Configuring the Remote Loader and Drivers in the *NetIQ Identity Manager Setup Guide for Windows*.

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.

Password Synchronization Support

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

Information Synchronized

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

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 Identity Manager server at the same time as the Identity Manager 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 3, "Creating a New Driver Object," on page 19) or upgrade an existing driver's configuration (see Chapter 5, "Upgrading an Existing Driver," on page 35).

You don't need to install the Identity Manager engine on this same machine. Using a Remote Loader, you can separate the engine and the driver shim, allowing you to balance the load on different machines or accommodate corporate directives.

The installation scenario you select determines how the driver shim is installed. If you choose to install the driver shim on the same machine as Identity Manager (where Identity Manager engine and Identity Vault are located), Identity Manager calls the driver shim directly. If you choose to install the driver shim on another machine, you must use the Remote Loader.

You can install the driver in one of the following ways:

- On a local machine: Install the JMS driver files on the Identity Manager server and connect to the JMS server by using the JMS PROVIDER URL (Connection Properties). See Installing Identity Manager in the *NetIQ Identity Manager Setup Guide for Linux* or Installing the Identity Vault in the *NetIQ Identity Manager Setup Guide for Windows*.

- On a remote machine, you can install in one of the following ways:

  - You can install the Identity Vault, the Identity Manager engine, and the driver on a separate computer from the JMS domain controller. This configuration leaves the domain controller free of any Identity Manager software.

  - Alternatively, you can install the Remote Loader and driver shim on the JMS domain controller, but install the Identity Vault and the Identity Manager engine on a separate server.

  See the instructions Configuring the Remote Loader and Drivers in the *NetIQ Identity Manager Setup Guide for Linux* or Configuring the Remote Loader and Drivers in the *NetIQ Identity Manager Setup Guide for Windows*.

3 Creating a New Driver Object

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 17) and you have configured the JMS system (see Chapter 4, "Configuring Messaging Vendors," on page 25), you can create the driver in the Identity Vault.

- "Creating the Driver in Designer" on page 19
- "Activating the Driver" on page 22
- "Adding Packages to an Existing Driver" on page 23

Creating the Driver in Designer

To create a JMS driver object, install the driver packages and then modify 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.

- "Importing the Current Driver Packages" on page 19
- "Installing the Driver Packages" on page 20
- "Configuring the Driver" on page 21
- "Deploying the Driver" on page 21
- "Starting the Driver" on page 22

NOTE: You should not create driver objects by using the new Identity Manager 4.0 and later configuration files through iManager. This method of creating driver objects is no longer supported. To create drivers, you now need to use the new package management features provided in Designer.

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**.
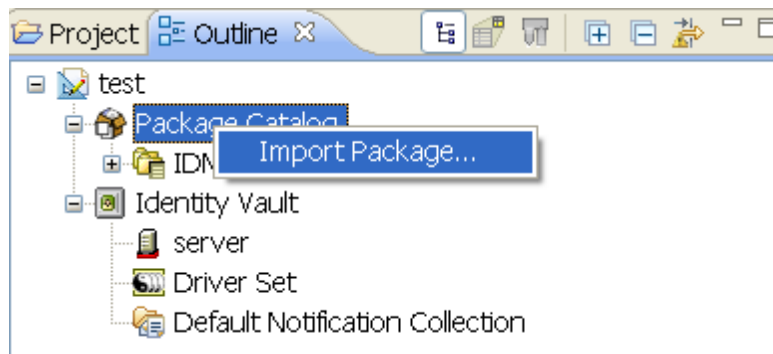
6 Select any JMS driver packages

   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 "Installing the Driver Packages" on page 20.

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. If you selected **Other**, then click **Next**. Otherwise, specify the Broker URL for the JMS vendor you selected, then click **Next**. 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.

5 On the Driver Information page, specify a name for the driver, then click **Next**.

6 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 Configuring the Remote Loader and Drivers in the *NetIQ Identity Manager Setup Guide for Linux* or Configuring the Remote Loader and Drivers in the *NetIQ Identity Manager Setup Guide for Windows*.

If you select **No**, skip to Step 7. If you select **Yes**, use the following information to complete the configuration of the Remote Loader, then click **Next**:

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 Identity Manager server. It must be the same password that is specified as the Driver Object Password on the Remote Loader.

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

8 After you have installed the driver, you must change the configuration for your environment. Proceed to "Configuring the Driver" on page 21.

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 *NetIQ Identity Manager - Using Designer to Create Policies* or *NetIQ Identity Manager Policies in iManager Guide*.

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

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.

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

7b Click **OK** twice.

For more information about defining a Security Equivalent User in objects for drivers in the Identity Vault, see "Establishing a Security Equivalent User" in the Identity Manager 4.5 Security Guide.

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**.

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 39.

Activating the Driver

The JMS driver is part of the Identity Manager Integration Module for Messaging.

This integration module requires a separate activation. After purchasing the integration module, you will receive activation details in your NetIQ Customer Center.

If you create a new JMS driver in a driver set that already includes an activated driver from this integration module, the new driver inherits the activation from the driver set.

If you create the driver in a driver set that has not been previously activated with this integration module, the driver will run in the evaluation mode for 90 days. You must activate the driver with this integration module during the evaluation period; otherwise, the driver will be disabled.

If driver activation has expired, the trace displays an error message indicating that you need to reactivate the driver to use it. For information on activation, refer to Activating Identity Manager in the *NetIQ Identity Manager Overview and Planning Guide*.

Adding Packages to an Existing Driver

You can add new functionality to an existing driver by adding new packages to an existing driver.

1 Right-click the driver, then click **Properties**.

2 Click **Packages**, then click the **Add Packages** icon ➕.

3 Select the packages to install. If the list is empty, there are no available packages to install.

4 (Optional) Deselect the **Show only applicable package versions** option, if you want to see all available packages for the driver, then click **OK**.

This option is only displayed on drivers. By default, only the packages that can be installed on the selected driver are displayed.

5 Click **Apply** to install all of the packages listed with the Install operation.

Package	Versi...	Upgra...	Operation
Password Synchronization Notificatio...	0.2.0		Select Operation...
Provisioning Notification Templates	0.2.0		Install
Password Management Notification T...	0.2.0		Install
Password Expiration Notification Tem...	0.2.0		Install
Job Default Notification Templates	0.2.0		Install
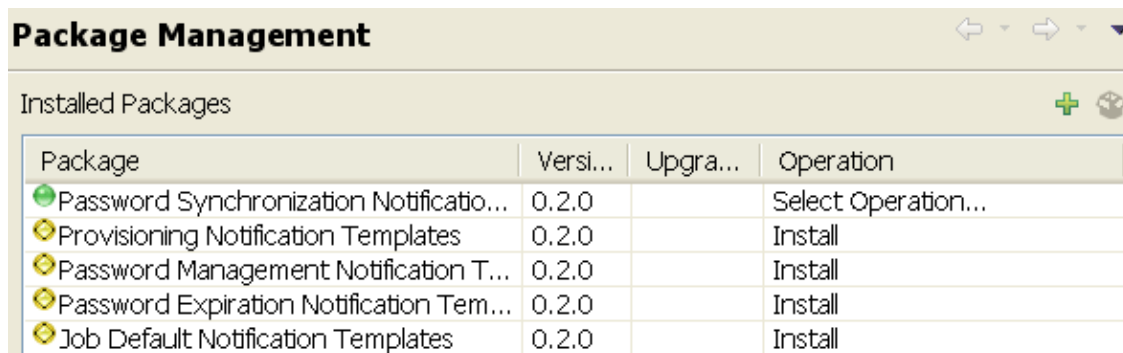
6 (Conditional) Fill in the fields with appropriate information to install the package you selected for the driver, then click **Next**.

7 Read the summary of the installation, then click **Finish**.

8 Click **OK** to close the Package Management page after you have reviewed the installed packages.

Package	Versi...	Upgra...	Operation
Job Default Notification Templates	0.2.0		Select Operation...
Password Expiration Notification Tem...	0.2.0		Select Operation...
Password Management Notification T...	0.2.0		Select Operation...
Password Synchronization Notificatio...	0.2.0		Select Operation...
Provisioning Notification Templates	0.2.0		Select Operation...
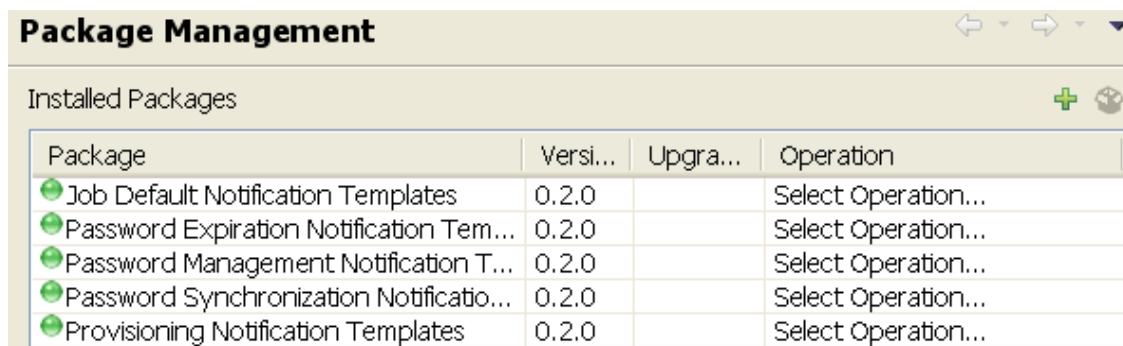
9 Repeat Step 1 through Step 8 for each driver where you want to add the new packages.

4 Configuring Messaging Vendors

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

- "Installing IBM MQ" on page 25
- "Installing on JBoss Messaging" on page 30
- "Installing on SonicMQ" on page 31
- "Installing on TIBCO EMS" on page 33

Installing IBM MQ

As part of installing IBM MQ 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.

- "Placing Prerequisite Jar Files and Scripts" on page 25
- "Creating a Server-Connection Channel and Queues" on page 26
- "Starting the Publish/Subscriber Broker" on page 27
- "Installing System Queues Necessary for Publish/Subscribe" on page 27
- "Creating a User Account" on page 27
- "Setting Up JMS for IBM MQ 8.x and 9.x" on page 28

Placing Prerequisite Jar Files and Scripts

1 (Conditional) If you are using IBM Websphere version lesser than 8.x, delete the following jars that are located at:

   Linux: `/opt/novell/eDirectory/lib/dirxml/classes`

   Windows: `C:\Novell\NDS\lib`

   - `com.ibm.mq.jar`
   - `com.ibm.mq.jmqi.jar`
   - `com.ibm.mqjms.jar`
   - `connector.jar`
   - `dhbcore.jar`
   - `fscontext.jar`
   - `jndi.jar`
   - `com.ibm.mq.commonservices.jar`
   - `com.ibm.mq.headers.jar`

2 Locate and copy the following jar files from your messaging server.

   For example, `<MQ_install_path>\java\lib`

   - `com.ibm.mq.commonservices.jar`
   - `com.ibm.mq.headers.jar`

- ◆ `com.ibm.mq.jar`
- ◆ `com.ibm.mq.jmqi.jar`
- ◆ `com.ibm.mqjms.jar`
- ◆ `fscontext.jar`

3 Download and copy `javax-jms-api-2.0.jar` from MVN Repository:

4 Paste the jar files that is mentioned in Step 2 and Step 3 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)

5 Restart eDirectory.

6 Locate the installation script that is saved 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`

7 Copy the following scripts to your messaging server at `<MQ_install_path>\java\bin`.
- ◆ `idm_jms_install.scp`
- ◆ `idm_jms_uninstall.scp`
- ◆ `idm_mq_install.mqsc`
- ◆ `idm_mq_uninstall.mqsc`
- ◆ `netiqsamplejms.prop`
- ◆ `vinstall.bat`
- ◆ `vuninstall.bat`

8 (Conditional) If necessary, restart your eDirectory server.

Creating a Server-Connection Channel and Queues

1 From the command line, change directories to `<MQ_HOME>\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 "Starting the Publish/Subscriber Broker" on page 27.

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 .

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 .

Creating a User Account

-
-

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 user name. 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

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 .

Setting Up JMS for IBM MQ 8.x and 9.x

1 On the MQ server, edit `<MQ_install_path>\java\bin\JMSAdmin.config`, and set the value to the following:

```
INITIAL_CONTEXT_FACTORY=com.sun.jndi.fscontext.RefFSContextFactory

PROVIDER_URL=file:/C:/JNDI-Directory
```

NOTE: Depending on your OS platform adjust the PROVIDER_URL to point to a valid file location path on the MQ server.

2 Edit `<MQ_install_path>\java\bin\JMSAdmin.bat` file, and set the JAVA envionment variable pointing to the correct location of the java executable under the jre folder set `JAVA="%MQ_JRE_PATH%\bin\java"`

3 Edit `<MQ_install_path>\java\bin\PSIVTRun.bat` file, and set the JAVA envionment variable pointing to the correct location of the java executable under the jre folder.

4 Edit netiqsamplejms.prop file, and set the values as follows:

Properties Key	Purpose
IDM_LOCAL_USER_ACCOUNT_NAME	A local windows user account will be created. This account will be used by the NetIQ JMS connetor to connect to MQ.
	Example: IDM
IDM_LOCAL_USER_ACCOUNT_PWD	Password for the local windows account.
	Example: NetIQ123
IDM_QUEUE_MANAGER_NAME	A queue manager is a program that provides messaging services to applications that use the Message Queue Interface to put and get messages from the queue. This is the name of the Queue Manager under which the IDM queues will be created and controlled.
	Example: IDM.QM
IDM_LISTENER_NAME	A listener is a WebSphere® MQ process that listens for connections to the queue manager. Provide the name for the listener that the above IDM queue manager will be using.
	Example: LISTENER.TCP
IDM_WEBMQ_HOST	This is the hostname or IP address of the server where MQ is running.
	Example: webmq.lab.com or <IP address>
IDM_LISTENER_PORT	This is the listener port where MQ server can accept messages. The default port for MQ is 1414, however each installation can be changed to listen on different ports. Enter the port number that is relevant to your installation.
	Example: 1414

Properties Key	Purpose
IDM_CHANNEL_NAME	A channel is a communication link used by distributed queue managers identified by a unique name.There are two categories of channel in WebSphere® MQ: ◆ Message channels: which are unidirectional, and transfer messages from one queue manager to another. ◆ MQI channels: which are bidirectional, and transfer MQI calls from a WebSphere MQ MQI client to a queue manager, and responses from a queue manager to a WebSphere MQ client. IDM integtration uses MQI channels. Provide a name that IDM will use. Example: CHANNEL.IDM.QM
IDM_DRV_QUEUE_NAME	A WebSphere MQ queue is a named object on which applications can put and get messages.Specify a name that uniquely identifies the IDM queue where IDM events are sent and received from. Example: IDM.EVENTSQ
IDM_DRV_TOPIC_NAME	A topic is the subject of the information that is published in a publish/subscribe message. Specify a unique TOPIC name that IDM can use to pub/sub messages. Example: IDM.EVENTSTOPIC
IDM_DRV_TOPIC_STRING	A publisher creates a message, labels it with a topic string that best fits the subject of the publication and then publishers it. To receive publications, a subscriber creates a subscription with a pattern matching topic string to select publication topics. Example: IDM.TOPICSTR

Properties Key	Purpose
IDM_JNDI_PROVIDER_URL	Sun engineered JNDI like JMS in that there is an API that MQ clients/apps use and an SPI or Service Provider Interface that is used by something called a "registry". Although JNDI can be implemented in LDAP, one of the base implementations that Sun provided right out of the box was to use the local filesystem as the registry. JNDI uses the .bindings file as the "registry" and it holds all the administered object definitions.

The objects in the .bindings file are represented in Name/Type/Value triplets. Each .bindings file typically has many objects. Each object has many attributes. Each attribute has a name, a value and the type of variable that holds the value. IBM provides the JMSAdmin tool to generate and read the .bindings file. But a valid file system location should be made availble for the JMSAdmin tool where it can create the .bindings file.

Example: C:\JNDI-Directory |

5 In command prompt change location to `<MQ_install_path>\java\bin\` and run vinstall.bat to create the MQ/JMS objects that will be used by the NetIQ JMS connector.

6 Make sure the `.bindings` file resides in the correct location.

The `.bindings` file is generated during the IBM 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, IBM MQ, Identity Manager 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 IBM MQ are on one server and the Identity Manager engine and Identity Vault are on another server (a Identity Manager server), copy the `.bindings` file to the Identity Manager server and make sure the PROVIDER_URL includes the correct path to the file. If multiple Identity Manager servers connect to the IBM MQ server, copy the `.bindings` file to the PROVIDER_URL path on each Identity Manager server.

7 Copy `.binding` file from IBM MQ server location (Example: C:\JNDI-Directory) to Identity Manager server. (Example: C:\JNDI-Directory).

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 jar files depending on your JBoss version:

For JBoss 4: Locate the following 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

For JBoss 6.2: Locate the following files:

- hornetq-core-client-2.3.3.final.jar
- hornetq-core.jar
- hornetq-jboss-as-integration-2.2.9.as7.final.jar
- jboss-client

NOTE: hornetq 2.4(latest)

Before using this jar, ensure you go through the known issue that exists with it in the JBoss Developer page (https://issues.jboss.org/browse/HORNETQ-1317).

---

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.

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.

- "Locating Prerequisite Jar Files" on page 32
- "Running Scripts to Configure the Messaging System" on page 32

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 .

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.

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.

- "Locating Prerequisite Client Jar Files" on page 33
- "Running Scripts to Configure the Messaging System" on page 33

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 "Running Scripts to Configure the Messaging System" on page 33.

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`
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`

5 Upgrading an Existing Driver

The following sections provide information to help you upgrade an existing driver to version 4.5:

- "Supported Upgrade Paths" on page 35
- "What's New in Version 4.0.0" on page 35
- "Upgrade Procedure" on page 35

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 or later is not supported.

What's New in Version 4.0.0

This version of the driver provides support for configuring Suite B communication between the Remote Loader and the Identity Manager engine. For more information, see "Authentication" on page 44. For more information about Suite B, see Suite B Cryptography.

Upgrade Procedure

The driver upgrade process involves updating the driver files and upgrading the installed driver packages. However, there is no separate procedure for updating the driver shim. The driver shim is updated with a new version of Identity Manager engine.

To upgrade the installed packages for the driver, perform the following actions in Designer:

1 Download the latest available packages.

To configure Designer to automatically read the package updates when a new version of a package is available, click **Windows** > **Preferences** > **NetIQ** > **Package Manager** > **Online Updates** in Designer. However, if you need to add a custom package to the Package Catalog, you can import the package `.jar` file. For more information about creating custom packages, see Developing Packages in *NetIQ Identity Manager Driver Administration Guide*.

2 Upgrade the installed packages.

  2a Open the project containing the driver.

  2b Right-click the driver for which you want to upgrade an installed package, then click **Driver** > **Properties**.

  2c Click **Packages**.

    If there is a newer version of a package, there is check mark displayed in the Upgrades column.

  2d Click **Select Operation** for the package that indicates there is an upgrade available.

  2e From the drop-down list, click **Upgrade**.

  2f Select the version that you want to upgrade to, then click **OK**.

2g Click **Apply**.

2h (Conditional) Fill in the fields with appropriate information to upgrade the package, then click**Next**.

2i Read the summary of the packages that will be installed, then click **Finish**.

2j Review the upgraded package, then click **OK** to close the Package Management page.

For detailed information, see "Upgrading the Identity Manager Drivers" in the *NetIQ Identity Manager Setup Guide for Linux* or "Upgrading the Identity Manager Drivers" in the *NetIQ Identity Manager Setup Guide for Windows*.

Applying the Driver Patch

The driver patch updates the driver files. You can install the patch as a `root` or `non-root` user.

Prerequisites

Before installing the patch, complete the following steps:

1 Take a back-up of the current driver configuration.

2 Stop the driver instance.

3 Stop the Identity Vault.

4 In a browser, navigate to the NetIQ Patch Finder Download Page.

5 Under **Patches**, click **Search Patches**.

6 Specify **Identity Manager** *nn* **Bidirectional eDirectory Driver nn** in the search box.

7 Download and unzip the contents of the patch file to a temporary location on your server.

Applying the Patch as a Root User

In a root installation, the driver patch installs the driver files in the default locations on Linux. On Windows, you need to manually copy the files to the default locations.

1 Delete or move the existing files that match the files being copied.

2 Update the driver files:

- **Linux:** Open a command prompt and run the following command to upgrade the existing RPM:

    `rpm -Uvh <Driver Patch File Temporary Location>/linux/novell-DXMLjms.rpm`

- **Windows:** Navigate to the `<Driver Patch File Temporary Location>\windows` folder and copy the `JMSShim.jar` file to `<IdentityManager installation>\NDS\lib` or `<IdentityManager installation>\RemoteLoader\<architecture>\lib` folder.

3 (Conditional) If the driver is running locally, start the Identity Vault and the driver instance.

For example, open a command prompt and run `ndsmanage startall`

4 (Conditional) If the driver is running with a Remote Loader instance, start the Remote Loader instance and the driver instance.

Applying the Patch as a Non-Root User

1 Verify that `<non-root eDirectory location>/rpm` directory exists and contains the file, `_db.000`.

The `_db.000` file is created during a non-root installation of the Identity Manager engine. Absence of this file in the directory might indicate that Identity Manager is not properly installed and it might be necessary to re-install it to place the file in the directory.

2 To set the `root` directory to non-`root` eDirectory location, enter the following command in the command prompt:

```
ROOTDIR=<non-root eDirectory location>
```

This will set the environmental variables to the directory where eDirectory is installed as a non-`root` user.

3 To install the driver files, enter the following command:

```
rpm --dbpath $ROOTDIR/rpm -Uvh --relocate=/usr=$ROOTDIR/opt/novell/eDirectory
--relocate=/etc=$ROOTDIR/etc --relocate=/opt/novell/eDirectory=$ROOTDIR/opt/
novell/eDirectory --relocate=/opt/novell/dirxml=$ROOTDIR/opt/novell/dirxml --
relocate=/var=$ROOTDIR/var --badreloc --nodeps --replacefiles <rpm-location>
```

For example, to install the JMS driver RPM, use this command:

```
rpm --dbpath $ROOTDIR/rpm -Uvh --relocate=/usr=$ROOTDIR/opt/novell/eDirectory
--relocate=/etc=$ROOTDIR/etc --relocate=/opt/novell/eDirectory=$ROOTDIR/opt/
novell/eDirectory --relocate=/opt/novell/dirxml=$ROOTDIR/opt/novell/dirxml --
relocate=/var=$ROOTDIR/var --badreloc --nodeps --replacefiles /home/user/
novell-DXMLjms.rpm
```

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 *NetIQ Identity Manager 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.

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

For information about configuring the driver to use DSTrace, see "Viewing Identity Manager Processes" in the *NetIQ Identity Manager Driver Administration Guide*.

The Driver Fails to Reconnect to WebMQ if Restarted

The JMS driver fails to reconnect with the following exception when connecting to either Web MQ 7.1 or 7.5.

```
Exception in thread "JMSCCThreadPoolMaster"
java.lang.IllegalThreadStateException
at java.lang.ThreadGroup.addUnstarted(Unknown Source)
at java.lang.Thread.init(Unknown Source)
at java.lang.Thread.init(Unknown Source)
at java.lang.Thread.<init>(Unknown Source)
at
com.ibm.msg.client.commonservices.j2se.workqueue.WorkQueueManagerImpl
ementation$ThreadPoolWorker.<init>(WorkQueueManagerImplementation.java:950)
at
com.ibm.msg.client.commonservices.j2se.workqueue.WorkQueueManagerImpl
ementation.createNewThread(WorkQueueManagerImplementation.java:496)
at
```

This issue is only observed with the latest third party Websphere JMS Client 7.5 and Websphere 7.1 jar files.

To workaround this issue, use the driver with Remote Loader.

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 *NetIQ Identity Manager 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 a Designer icon.

- "Driver Configuration" on page 43
- "Global Configuration Values" on page 52

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:

- "Driver Module" on page 43
- "Driver Object Password" on page 44
- "Authentication" on page 44
- "Startup Option" on page 45
- "Driver Parameters" on page 45
- "ECMAScript" on page 51
- "Global Configurations" on page 52

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.

Driver Object Password

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.

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.

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

Remote Loader Connection Parameter: Used only if the driver is connecting to the application through the Remote Loader.

In iManager, enter `hostname=xxx.xxx.xxx.xxx port=xxxx secureprotocol=TLS version enforceSuiteB=`*true/false* `kmo=certificatename`.

- `hostname` specifies the IP address of the Remote Loader server.

- `port` specifies the TCP/IP port on which the Remote Loader listens for connections from the remote interface shim. The default port for the Remote Loader is 8090.

- `secureprotocol` specifies the version of the TLS protocol that the Remote Loader uses to connect to the Identity Manager engine. Identity Manager supports TLSv1, TLS v1_1, and TLSv1_2 versions only.

- `enforceSuiteB` specifies whether the Remote Loader uses Suite B for communicating with the Identity Manager engine. To use Suite B, specify `enforceSuiteB=true`. The communication supports only TLS version 1.2 version. Communication is not established if the connection has non-Suite B authentication algorithms.

- The `kmo` entry is optional. Use it only when an SSL connection exists between the Remote Loader and the Identity Manager engine.

  For example: `hostname=10.0.0.1 port=8090 kmo=IDMCertificate`

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

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.

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

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. This contains .binding.file. For example: `C:\JNDI-Directory`

- **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.

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.

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.

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**.

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	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 "Viewing Identity Manager Processes" in the *NetIQ Identity Manager Driver Administration Guide*.