Novell Identity Manager Driver for Java* Messaging Service

3.5.1

www.novell.com

DRIVER GUIDE

September 28, 2007

Novell®

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. Please refer to www.novell.com/info/exports/ for more information on exporting Novell software. Novell assumes no responsibility for your failure to obtain any necessary export approvals.

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 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

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

Novell Trademarks

Novell is a registered trademark of Novell, Inc., in the United States and other countries.
SUSE is a registered trademark of Novell, Inc., in the United States and other countries.

Third-Party Materials

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

Contents

About This Guide

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

Audience

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

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation, or go to www.novell.com/documentation/feedback.html and enter your comments there.

Documentation Updates

For the most recent version of the *Identity Manager Driver Guide for Java Messaging Service*, visit the Identity Manager Driver Web site (http://www.novell.com/documentation/idmdrivers).

Additional Documentation

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

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

1.1 What’s New

In this section:

1.1.1 Driver Features

In this release of the JMS driver, we have added support for the TIBCO* EMS message broker.

1.1.2 Identity Manager Features

For information about the new features in Identity Manager, see “What's New in Identity Manager 3.5.1? ” in the*Identity Manager 3.5.1 Installation Guide*.

1.2 Key Driver Features

The sections below contains a list of the key driver features.

1.2.1 Local Platforms

The JMS driver can be installed locally on any of the supported Identity Manager platforms.

1.2.2 Remote Platforms

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

For more information about installing the Remote Loader services, see "Installing the Remote Loader" in the *Novell Identity Manager 3.5.1 Administration Guide*.

1.2.3 Role-Based Entitlements

The JMS driver does not have Role-Based Entitlement functionality defined with the example configuration files. The driver does support entitlements, if there are policies created for the driver to consume.

1.2.4 Password Synchronization Support

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

1.2.5 Information Synchronized

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

1.3 Key Terminology

The Identity Manager Driver for Java Messaging Service (JMS), hereafter referred to as the driver, allows 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 terms are used throughout this document, so it's important to define how they are used:

- **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:** A message broker is 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 once and 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.)

1.4 JMS Messaging Models

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

- Section 1.4.1, “Point-to-Point Messaging,” on page 13
- Section 1.4.2, “Publish/Subscribe Messaging,” on page 13

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*

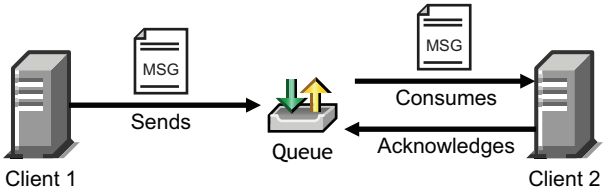
Abstract Terminology	Point-to-Point Terminology	Publish/Subscribe Terminology
Destination	Queue	Topic
Sender (or Producer)	Sender	Publisher
Receiver (or Consumer)	Receiver	Subscriber

1.4.1 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*

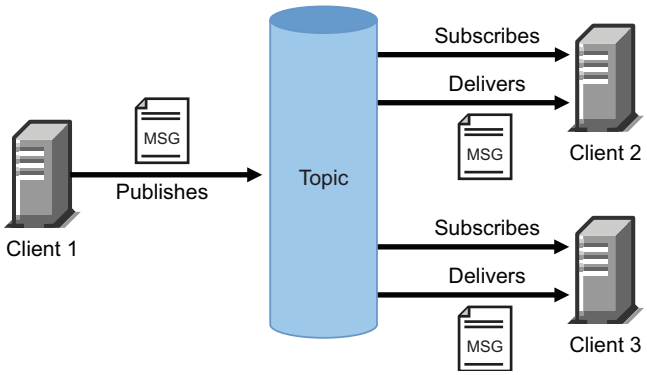


1.4.2 Publish/Subscribe Messaging

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*



1.5 JMS Messages

This section contains information about JMS message structures and message types, as well as examples for each.

- Section 1.5.1, “Message Structure,” on page 14
- Section 1.5.2, “Message Types,” on page 15

1.5.1 Message Structure

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

Example JMS Message Envelope

```
<jms:message xmlns:jms="urn:idm:jms">
<jms:headers>
      <!-- standard JMS headers start with "JMS" -->
      <!-- client-assignable headers -->
      <jms:header jms:name="JMSDeliveryMode"/>
      <jms:header jms:name="JMSExpiration"/>
      <jms:header jms:name="JMSPriority"/
      <jms:header jms:name="JMSReplyTo"/>
      <jms:header jms:name="JMSCorrelationID"/>
      <jms:header jms:name="JMSType"/>
   </jms:headers>
   <jms:properties>
      <!-- standard JMS properties start with "JMSX" -->
      <jms:property jms:name="JMSXUserID"/>
      <jms:property jms:name="JMSXAppID"/>
      <jms:property jms:name="JMSXProducerTXID"/>
      <jms:property jms:name="JMSXConsumerTXID"/>
      <jms:property jms:name="JMSXRcvTimestamp"/>
```

```
      <jms:property jms:name="JMSXDeliveryCount"/>
      <jms:property jms:name="JMSXState"/>
      <jms:property jms:name="JMSXGroupID"/>
      <jms:property jms:name="JMSXGroupSeq"/>
      <!-- provider-specific properties start with "JMS_" -->
     <!-- application-specific properties start with anything else -->
   </jms:properties>
  <jms:body/>
</jms:message>
```

1.5.2 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>
```

1.6 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.

1.6.1 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 IDM 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>
```

1.6.2 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.

1.7 Additional Resources

For more information about JMS and Messaging Models, see the following Web sites:

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

Before Installing the Driver

2

Before installing the driver, ensure that you meet the driver prerequisites and are using a supported vendor.

2.1 Driver Prerequisites

The Identity Manager Driver for JMS requires the following:

- ❑ Novell® iManager 2.6 or later installed on the server
- ❑ Novell Identity Manager 3.5.1 installed on the server
- ❑ JMS 1.1 or 1.0.2
- ❑ A supported JMS vendor

2.2 Supported Platforms and JMS Vendors

The driver runs on all Identity Manager-enabled platforms, including Windows* NT*/2000, NetWare®, Solaris*, Linux, and AIX*.

The driver supports the following vendors:

- JBossMQ v4.*x*
- IBM* WebSphere* MQ v6.*x*
- SonicMQ* v7.*x*
- TIBCO EMS v4

Designing the Driver

3

Identity Manager contains a visual configuration tool that provides a simple yet powerful way to design and configure Identity Manager projects. Designer for Identity Manager allows you to:

- Graphically model the implementation
- Re-use configurations to help reduce deployment time
- Create and test scenarios to ensure proper policy definition before deploying in production
- Automatically generate project documentation for all implementation details
- Use the offline mode to safely configure implementations outside of the production environment
- Design and manage policies

Drivers can be deployed through Designer or iManager. Novell® recommends that you use Designer to configure and test the drivers, and that you use iManager for administration after drivers are deployed into your environment. For more information about Designer, see the *Designer 2.1 for Identity Manager 3.5.1*.

Installing the Driver

4

The driver installation process adds the driver configuration files to the Identity Manager server, however, it does not create an object in the Identity Vault. To create the driver object, see Chapter 5, “Configuring the Driver,” on page 25.

4.1 Installing the Driver Locally

If you are going to install the driver locally, the server that runs the driver is required to have the following installed:

- eDirectory™ 8.73 SP8 or above, or 8.8 SP1 or above
- JMS 1.1 or 1.0.2
- A supported JMS vendor

The JMS driver is installed during the installation of Identity Manager. See “Installing Identity Manager” in the*Identity Manager 3.5.1 Installation Guide* for the installation instructions.

4.2 Installing the Driver Remotely

If you do not want to install eDirectory and Identity Manager on the server that has JMS or a supported JMS vendor installed on it, use the Remote Loader service. The Remote Loader service is installed and then connects to the server running Identity Manager. For installation instructions see, “Deciding Whether to Use the Remote Loader” in the *Novell Identity Manager 3.5.1 Administration Guide*.

Configuring the Driver

5

After the driver installation is complete, the driver needs to be created and configured. The driver is created through the Driver Configuration Wizard that is launched from Designer or iManager.

5.1 Configuring the Driver in Designer

Designer allows you to import the basic driver configuration file for the JMS driver. This file creates and configures the objects and policies needed to make the driver work properly. There are many different ways of importing the driver configuration file; the following documents one method of creating the driver and importing the driver’s configuration.

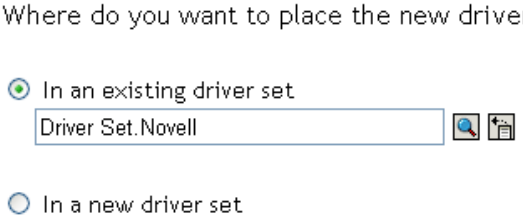
1. Open a project in Designer, go to the Modeler, right-click the Driver Set object, then select *New > Driver*.
2. From the drop-down list, select *IDM Driver for JMS*, then click *Run*.
3. Configure the driver by filling in the fields.

   Provide information specific to your environment. For information on the settings, see Section 5.3, “Driver Configuration Options,” on page 27.
4. After specifying the parameters, click *Finish* to import the driver.
5. Customize and test the driver before deploying the driver into the production environment.
6. After the driver is fully tested, deploy the driver into the Identity Vault as described in “Deploying a Driver to an Identity Vault” in the *Designer 2.1 for Identity Manager 3.5.1*.

5.2 Configuring the Driver in iManager

The JMS driver configuration file is a sample configuration. You installed this file when you installed the Identity Manager Web components on an iManager server. Think of the driver configuration as a template that you import and customize or configure for your environment.

1. In iManager, select *Identity Manager Utilities > Import Configurations*.
2. Select an existing driver set or select a new driver set.

3. If you selected an existing driver set, continue with Step 4.

or

If you selected to place the driver in a new driver set, skip to Step 6.

4 If you selected an existing driver set, browse to and select the driver set, then click *Next*.

5 Browse to and select the server the driver is associated with, click *Next*, then skip to Step 7.

6 If you selected to place the driver in a new driver set, click *Next*.

7 Define the properties of the new driver set, then click *Next*.

7a Specify the name of the driver set.

7b Browse to and select the context where the driver set is created.

7c Browse to and select the server you want the driver set associated with.

7d Leave the *Create a new partition on this driver set* option selected.

7e Click *Next*.

Novell recommends that you create a partition for the driver object. For Identity Manager to function, the server that is associated with the driver set must hold a real replica of the Identity Manager objects. If the server holds a Master or Read/Write replica of the context where the objects are to be created, then the partition is not required.

8 Select how you want the driver configurations sorted:

- All configurations
- Identity Manager 3.5 configurations
- Identity Manager 3.0 configurations
- Configurations not associated with an IDM version

9 Select the JMS driver, then click *Next*.

10 Configure the driver by filling in the configuration parameters, then click *Next*.

For information on the settings, see Section 5.3, “Driver Configuration Options,” on page 27.

11 Select *Define Security Equivalences*.

11a Click *Add*, then browse to and select a user object that has the rights the driver needs to have on the server.

Many administrators use the Administrator User object in the Identity Vault for this task. However, you might want to create another object, such as a DriversUser, 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.

11b Click *OK* twice.

12 Select *Exclude Administrative Roles*.

12a Click *Add*, then browse to and select all objects that represent administrative roles and exclude them from replication with the driver.

Exclude the security-equivalence object in the Identity Vault (for example, DriversUser) that you specified in Step 11. If you delete the security-equivalence object, you have removed the rights from the driver. Therefore, the driver can’t make changes to Identity Manager.

If there are objects that are currently excluded, they do not appear in the Excluded users list unless you select *Retrieve Current Exclusions*.

12b Click *OK* twice.

13 Click *Next*.

14 View the summary, then click *Finish*.

15 After configuring the driver in iManager, proceed with Chapter 6, "Configuring Messaging Vendors," on page 31. You can also view information about configuring additional driver properties in Chapter 10, "Customizing the Driver," on page 65.

5.3 Driver Configuration Options

The following table explains the parameters you must provide during initial driver configuration.

NOTE: The options are presented on multiple screens and some options are displayed only if the answer to a previous prompt requires more information to properly configure the policy, such as if you choose a remote installation vs. local installation.

Table 5-1 *Configuration Options for the JMS Driver*

Import Prompt	Description
Driver name	The name of the driver contained in the driver configuration file is JMS3. Specify the actual name you want to use for the driver.
Target JMS provider	Specify which target JMS provider you want to use: ◆ IBM WebSphere MQ 6 ◆ JBossMQ 4 ◆ SonicMQ 7 ◆ Tibco EMS 4
Dataflow	The dataflow type that you want. **Bi-directional:** Specifies that the JMS provider and Identity Manager are both authoritative data sources. **Identity Manager to JMS provider (subscribe):** Specifies that Identity Manager is the authoritative data source. **JMS provider to Identity Manager (publish):** Specifies that the JMS provider is the authoritative data source.
Local or Remote	Configure the driver for use with the Remote Loader service by selecting Remote, or select Local to configure the driver for local use. If Local is selected, some of the remaining prompts are not displayed.
IP Address of Target JMS Broker	Specify the IP address of the target JMS broker in the following format: `123.456.789.123`.
Broker port number	The port number for your broker. The default value is 1099.
Destination type	Specify the type of destination you want to use.
Remote Host Name and Port	For remote driver configuration only. Specify the host name or IP address and port number where the Remote Loader Service has been installed and is running for this driver. The default port is 8090.

Import Prompt	Description
Remote Password	For remote driver configuration only. The Remote Loader password is used to control access to the Remote Loader instance. It must be the same password that is specified as the Remote Loader password on the Identity Manager Remote Loader.
Driver Password	For remote driver configuration only. The driver object password is used by the Remote Loader to authenticate to the Identity Manager server. It must be the same password that is specified in the Driver Object Password field on the Identity Manager Remote Loader.

5.4 Starting the Driver

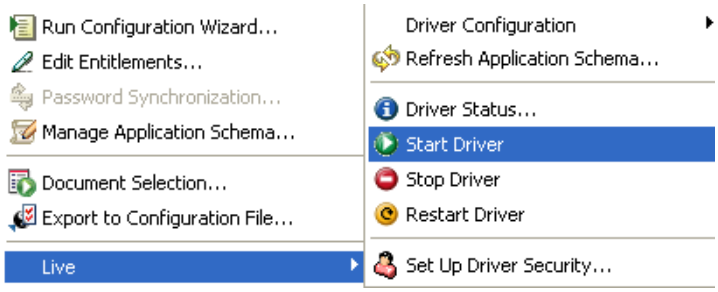
You must start the driver for information to be synchronized. Synchronization takes place on an object-by-object basis as changes are made to individual objects. After the driver is started, the objects are synchronized only if they are modified or when a new object is created. If you want to have the data immediately synchronize, you must initiate that process as explained in Chapter 8, "Synchronizing Objects," on page 43.

The driver is started through Designer or iManager.

- Section 5.4.1, "Starting the Driver in Designer," on page 28
- Section 5.4.2, "Starting the Driver in iManager," on page 28

5.4.1 Starting the Driver in Designer

1. Open a project in Designer, go to the Modeler, then right-click the connection between the application and the Driver Set object.
2. Select *Live* > *Start Driver*.

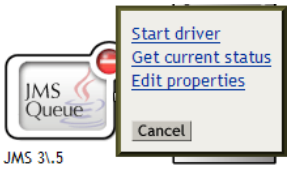


5.4.2 Starting the Driver in iManager

1. Click *Identity Manager* > *Identity Manager Overview*.
2. Select *Search entire tree* or *Search in container* for the Driver Set object where the driver is installed, then click *Search*.

   2a If you selected *Search in container*, browse to and select the container where you want to search for the Driver Set object, then click *Search*.

3 Click the upper right corner of the icon for the driver you want to start, then click *Start driver*.



Configuring Messaging Vendors

6

6.1 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.

6.1.1 Placing Prerequisite Jar Files and Scripts

1. On your messaging server, locate the following jar files:
   - `com.ibm.mq.jar`
   - `com.ibm.mq.pcf.jar`
   - `com.ibm.mqjms.jar`
   - `connect.jar`
   - `dhbcore.jar`
   - `jta.jar`
   - `mqcontext.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`
NetWare®	Local installation: `sys:\system\lib`

Platform	Directory Path
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`
NetWare	`sys:\System\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.

6.1.2 Create a Default Queue Manager

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

Manually Create 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.

Set 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.

6.1.3 Create 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.

6.1.4 Start 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.

6.1.5 Install 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."

6.1.6 Create a User Account

To create a User:

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

2 Expand *Local Users and Groups* subtree.

3 Right-click the *Users* folder > 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.

To make the User a member of the mqm Group

1 Right-click the newly created user > Select *Properties*.

2 Select the *Member Of* tab.

3 Left-click the mqm group

4 Click *Add*.

5 Click *OK* twice.

6.1.7 Setting Up JMS

Download the following support packs:

- ME01 (`mqcontext.jar`) = InitialContextFactory (http://www-1.ibm.com/support/docview.wss?rs=171&uid=swg24004684&loc=en_US&cs=utf-8&lang=en)
- MS0B (`com.ibm.mq.pcf.jar`) = Java classes for PCF (http://www-1.ibm.com/support/docview.wss?rs=171&uid=swg24000668&loc=en_US&cs=utf-8&lang=en)

1 Decompress the support packs and copy the `mqcontext.jar` and `com.ibm.mq.pcf.jar` files to the Program Files\IBM\WebSphere MQ\Java\lib directory.

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

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

::add these lines before call to java
set CLASSPATH=%CLASSPATH%;%MQ_JAVA_LIB_PATH%\com.ibm.mq.pcf.jar
set CLASSPATH=%CLASSPATH%;%MQ_JAVA_LIB_PATH%\mqcontext.jar
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
```

3 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.ibm.mq.jms.context.WMQInitialContextFa
ctory

# comment out all PROVIDER_URL lines and add this one:
PROVIDER_URL=localhost:1414/SYSTEM.DEF.SVRCONN
```

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

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

5 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

6 Update the connection factory IP addresses and port in idm_jms_install.scp.

7 Update the listener port in idm_mq_install.mqsc.

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

9 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.

10 From the command line, manually start the publish/subscribe broker by executing the following:

Program Files\IBM\WebSphere MQ\bin\strmqbrk.exe.

11 From the command line, ensure the publish/subscribe broker is configured correctly by executing the following:

Program Files\IBM\WebSphere MQ\Java\PSIVTRun.bat -nojndi -t

6.2 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.

6.2.1 Locate Prerequisite Jar Files

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
NetWare	Local installation: sys:\system\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.

6.3 Installing on SonicMQ

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

- Section 6.3.1, "Locate Prerequisite Jar Files," on page 36
- Section 6.3.2, "Running Scripts to Configure the Messaging System," on page 37

6.3.1 Locate 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_SSC.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`
NetWare	Local installation: `sys:\system\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.

6.3.2 Running Scripts to Configure the Messaging System

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

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

Platform	Directory Path
Windows	`C:\Novell\NDS\DirXMLUtilities\jms\sonic`
NetWare	`sys:\System\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.

6.4 Installing on TIBCO EMS

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

- Section 6.3.1, “Locate Prerequisite Jar Files,” on page 36
- Section 6.3.2, “Running Scripts to Configure the Messaging System,” on page 37

6.4.1 Locate 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
NetWare	Local installation: sys:\system\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.

6.4.2 Running Scripts to Configure the Messaging System

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

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

Platform	Directory Path
Windows	C:\Novell\NDS\DirXMLUtilities\jms\tibco_ems
NetWare	sys:\System\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
```

Activating the Driver

7

Novell® Identity Manager, Integration Modules, and the Provisioning Module must be activated within 90 days of installation, or they shut down. At any time during the 90 days, or afterward, you can choose to activate Identity Manager products.

To activate the driver, see Activating Novell Identity Manager Products (http://www.novell.com/documentation/idm35/install/data/afbx4oc.html).

Synchronizing Objects

8

This section explains driver and object synchronization in DirXML® 1.1a, Identity Manager 2.0, and Identity Manager 3.0. Driver synchronization was not available for DirXML 1.0 and DirXML 1.1.

After the driver is created, instead of waiting for objects to be modified or created, the data between the two connected systems can be sent through the synchronization process.

8.1 What Is Synchronization?

The actions commonly referred to as “synchronization” in Identity Manager refer to several different but related actions:

- Synchronization (or merging) of attribute values of an object in the Identity Vault with the corresponding attribute values of an associated object in a connected system.
- Migration of an Identity Vault object that does not have an associated connected system object into the connected system (that is, the creation of a new object in the connected system).
- Generation of the list of objects to submit to the driver’s Subscriber channel for synchronization or migration in response to a user request (a manual synchronization).
- Generation of the list of objects to submit to the driver’s Subscriber channel for synchronization or migration in response to enabling a formerly disabled driver, or in response to a cache error.

8.2 When Is Synchronization Done?

The Metadirectory engine performs object synchronization or merging in the follow circumstances:

- A <sync> event element is submitted on the Subscriber or Publisher channel.
- A <sync> event element is submitted on the Subscriber channel in the following circumstances:
  - The state of the object’s association value is set to “manual” or “migrate.” (This causes an eDirectory™ event, which in turn causes the Identity Manager caching system to queue an object synchronization command in the affected driver’s cache.)
  - An object synchronization command is read from the driver’s cache.
- A <sync> event element is submitted on the Publisher channel in the following circumstances:
  - A driver submits a <sync> event element. No known driver currently does this.

- The Metadirectory engine submits a <sync> event element for each object found as the result of a migrate-into-NDS query. These <sync> events are submitted using the Subscriber thread, but are processed using the Publisher channel filter and policies.

- An <add> event (real or synthetic) is submitted on a channel and the channel Matching policy finds a matching object in the target system.
- An <add> event with an association is submitted on the Subscriber channel. This normally occurs only in exceptional cases, such as the bulk load of objects into eDirectory with DirXML-Associations attribute values.
- An <add> event is submitted on the Publisher channel and an object is found in eDirectory that already has the association value reported with the <add> event.

The Metadirectory engine generates synchronization requests for zero or more objects in the following cases:

- The user issues a manual driver synchronization request. This corresponds to the *Resync* button in the Driver Set property page in ConsoleOne®, or to the *Synchronize* button on the iManager Identity Manager Driver Overview page.
- The Metadirectory engine encounters an error with the driver's cache and cannot recover from the cache error. The driver's cache is deleted and the engine generates object synchronization commands as detailed in Section 8.3, "How Does the Metadirectory Engine Decide Which Object to Synchronize?," on page 44.

8.3 How Does the Metadirectory Engine Decide Which Object to Synchronize?

The Metadirectory engine processes both manually initiated and automatically initiated synchronization requests in the same manner. The only difference in the processing of manually initiated versus automatically initiated driver synchronization requests is the starting filter time used to filter objects being considered for synchronization.

The starting filter time is used to filter objects that have modification or creation times that are older than the starting time specified with the synchronization request.

For automatically initiated driver synchronization, the starting filter time is obtained from the time stamps of cached eDirectory events. In particular, the starting filter time is the earliest time for the events cached that haven't yet been successfully processed by the driver's Subscriber channel.

For manually initiated driver synchronization, the default starting filter time is the earliest time in the eDirectory database. In Identity Manager 2 and Identity Manager 3, an explicit starting filter time can also be set. In DirXML 1.1a there is no facility to set the starting filter time value for synchronization when manually initiating driver synchronization.

The Metadirectory engine creates a list of objects to be synchronized on the Subscriber channel in the following manner:

1. It finds all objects that:
   - Have an entry modification time stamp greater than or equal to the starting filter time

     and
   - Have a DirXML-Associations attribute value that references the driver being synchronized.

2. It finds all objects that have an entry creation time stamp greater than or equal to the starting filter time.
3. It adds a synchronize object command to the driver cache for each unique object found that has an entry modification time stamp greater than or equal to the starting filter time and has a DirXML-Association attribute value that references the driver being synchronized.

8.4 How Does Synchronization Work?

After the Metadirectory engine determines that an object is to be synchronized, the following processes occur:

1. Each system (the Identity Vault and the connected system) is queried for all attribute values in the appropriate filters.
   - eDirectory is queried for all values in the Subscriber filter, and for values that are marked for synchronization in Identity Manager 2 and Identity Manager 3.
   - The connected system is queried for all values in the Publisher filter, and for values that are marked for synchronization if Identity Manager 2 and Identity Manager 3.
2. The returned attribute values are compared and modification lists are prepared for the Identity Vault and the connected system according to Table 8-1 on page 46, Table 8-2 on page 47, and Table 8-3 on page 49.

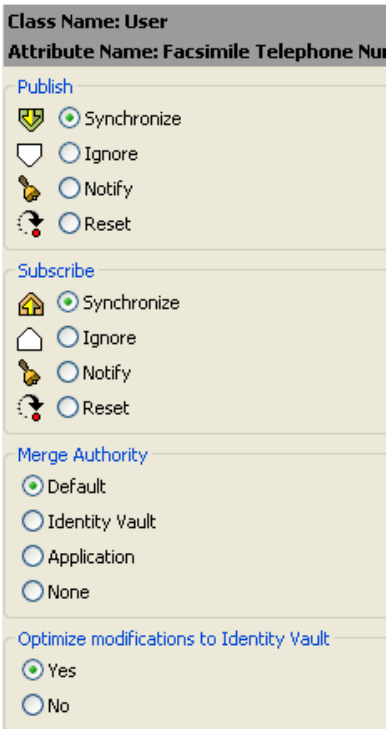
   In the tables the following pseudo-equations are used:
   - “Left = Right” indicates that the left side receives all values from the right side.
   - “Left = Right[1]” indicates that the left side receives one value from the right side. If there is more than one value, is indeterminate.
   - “Left += Right” indicates that the left side adds the right side values to the left side’s existing values.
   - “Left = Left + Right” indicates that the left sides receives the union of the values of the left and right sides.

There are three different combinations of selected items in the filter, they create different outputs.

8.4.1 Scenario One

The attribute is set to synchronize on the Publisher and Subscriber channels and merge authority is set to default.

Figure 8-1 *Scenario One*



The following table contains the values that the Metadirectory engine synchronizes when the attribute is sent through a filter that is set to the configuration for scenario one. The table shows different outputs depending upon whether the attribute comes from the Identity Vault or the Application, if the attribute is single-valued or multi-valued, and if the attribute is empty or non-empty.

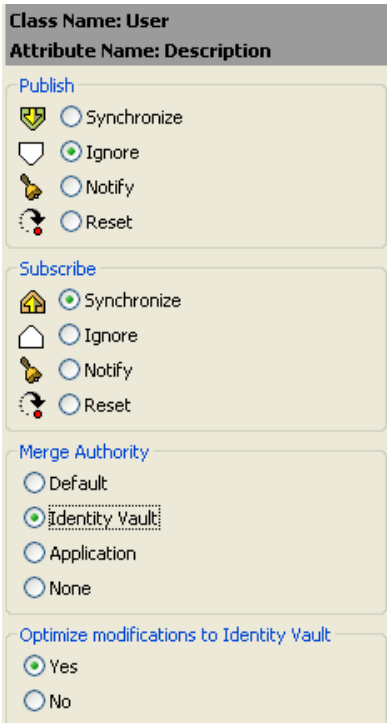
Table 8-1 *Output of Scenario One*

	Identity Vault single-valued empty	**Identity Vault single-valued non-empty**	**Identity Vault multi-valued empty**	**Identity Vault multi-valued non-empty**
Application single-valued empty	No change	App = Identity Vault	No change	App = Identity Vault[1]
Application single-valued non-empty	Identity Vault = App	App = Identity Vault	Identity Vault = App	Identity Vault + = App
Application multi-valued empty	No change	App = Identity Vault	No change	App = Identity Vault
Application multi-valued non-empty	Identity Vault = App[1]	App + = Identity Vault	Identity Vault = App	App = App + Identity Vault Identity Vault = App + Identity Vault

8.4.2 Scenario Two

The attribute is set to synchronize only on the Subscriber channel, or it is set to synchronize on both the Subscriber and Publisher channels and the merge authority is set to the Identity Vault.

Figure 8-2 *Scenario Two*



The following table contains the values that the Metadirectory engine synchronizes when the attribute is sent through a filter that is set to the configuration for scenario Two. The table shows different outputs depending upon whether the attribute comes from the Identity Vault or the Application, if the attribute is single-valued or multi-valued, and if the attribute is empty or non-empty.

Table 8-2 *Output of Scenario Two*

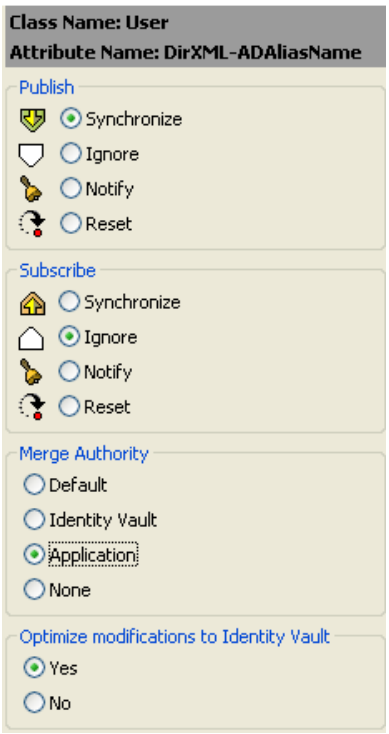
	Identity Vault single-valued empty	**Identity Vault single-valued non-empty**	**Identity Vault multi-valued empty**	**Identity Vault multi-valued non-empty**
Application single-valued empty	No change	App = Identity Vault	No change	App = Identity Vault[1]
Application single-valued empty	App = empty	App = Identity Vault	Identity Vault = App	App = Identity Vault[1]
Application multi-valued empty	No change	App = Identity Vault	No change	App = Identity Vault

	Identity Vault single-valued empty	Identity Vault single-valued non-empty	Identity Vault multi-valued empty	Identity Vault multi-valued non-empty
Application multi-valued non-empty	App = empty	App = Identity Vault	App = empty	App = Identity Vault

8.4.3 Scenario Three

The attribute is set to synchronize on the Publisher channel or the merge authority is set to application.

Figure 8-3 *Scenario Three*



The following table contains the values that the Metadirectory engine synchronizes when the attribute is sent through a filter that is set to the configuration for scenario Three. The table shows different outputs depending upon whether the attribute comes from the Identity Vault or the Application, if the attribute is single-valued or multi-valued, and if the attribute is empty or non-empty.

Table 8-3 *Output of Scenario Three*

	Identity Vault single-valued empty	**Identity Vault single-valued non-empty**	**Identity Vault multi-valued empty**	**Identity Vault multi-valued non-empty**
Application single-valued empty	No change	Identity Vault = empty	No change	Identity Vault = empty
Application single-valued non-empty	Identity Vault = App	Identity Vault = App	Identity Vault = App	Identity Vault = App
Application multi-valued empty	No change	Identity Vault = empty	No change	Identity Vault = empty
Application multi-valued non-empty	Identity Vault = App[1]	Identity Vault = App[1]	Identity Vault = App	Identity Vault = App

Managing the Driver

9

The driver can be managed through Designer, iManager or the DirXML® Command Line utility.

9.1 Starting, Stopping, or Restarting the Driver

9.1.1 Starting the Driver in Designer

1. Open a project in the Modeler, then right-click the driver line.
2. Select *Live* > *Start Driver*.

9.1.2 Starting the Driver in iManager

1. In iManager, click *Identity Manager* > *Identity Manager Overview*.
2. Browse to the driver set where the driver exists, then click *Search*.
3. Click the upper right corner of the driver icon, then click *Start driver*.

9.1.3 Stopping the Driver in Designer

1. Open a project in the Modeler, then right-click the driver line.
2. Select *Live* > *Stop Driver*.

9.1.4 Stopping the Driver in iManager

1. In iManager, click *Identity Manager* > *Identity Manager Overview*.
2. Browse to the driver set where the driver exists, then click *Search*.

3 Click the upper right corner of the driver icon, then click *Stop driver*.

9.1.5 Restarting the Driver in Designer

1 Open a project in the Modeler, then right-click the driver line.
2 Select *Live* > *Restart Driver*.

9.1.6 Restarting the Driver in iManager

1 In iManager, click *Identity Manager* > *Identity Manager Overview*.
2 Browse to the driver set where the driver exists, then click *Search*.
3 Click the upper right corner of the driver icon, then click *Restart driver*.

9.2 Using the DirXML Command Line Utility

The DirXML Command Line utility provides command line access to manage the driver. This utility is not a replacement for iManager or Designer. The primary use of this utility is to allow you to create platform-specific scripts to manage the driver.

For example, you could create a shell script on Linux to check the status of the driver. See Appendix A, "DirXML Command Line Utility," on page 89 for detailed information about the DirXML Command Line utility. For daily tasks, use iManager or Designer.

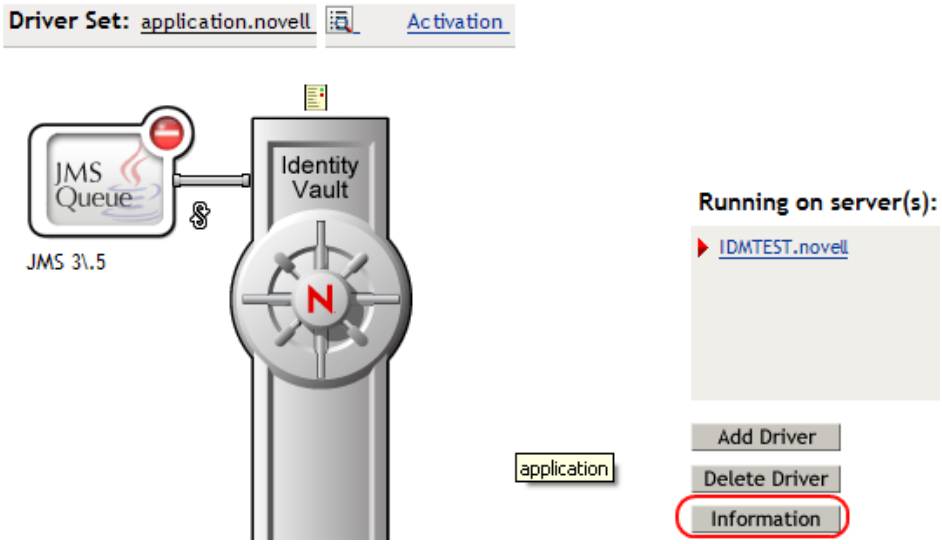
9.3 Viewing Driver Versioning Information

The Versioning Discovery tool only exists in iManager.

9.3.1 Viewing a Hierarchical Display of Versioning Information

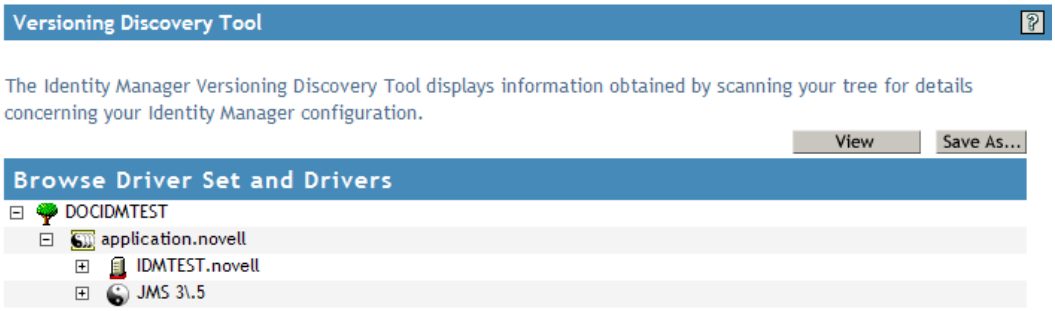
1 To find your Driver Set object in iManager, click *Identity Manager* > *Identity Manager Overview*, then click *Search*.

2 In the Identity Manager Overview, click *Information.*



You can also select *Identity Manager Utilities* > *Versions Discovery*, browse to and select the Driver Set object, then click *OK*.

3 View a top-level or unexpanded display of versioning information.

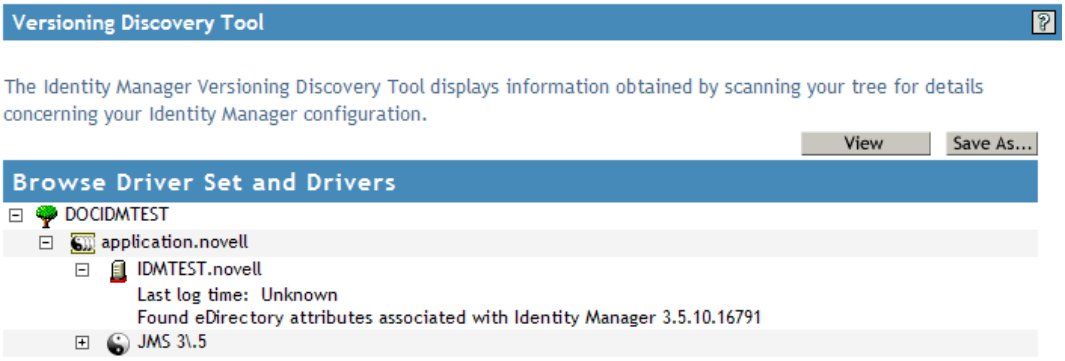


The unexpanded hierarchical view displays the following:

- The eDirectory™ tree that you are authenticated to
- The Driver Set object that you selected
- Servers that are associated with the Driver Set object

  If the Driver Set object is associated with two or more servers, you can view Identity Manager information on each server.
- Drivers

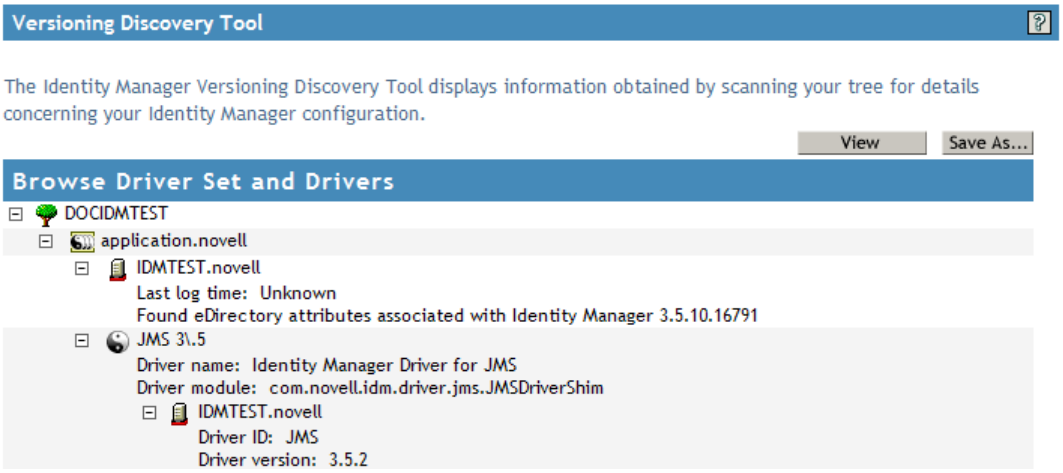
4 View versioning information related to servers by expanding the server icon.



The expanded view of a top-level server icon displays the following:

- Last log time
- Version of Identity Manager that is running on the server

5 View versioning information related to drivers by expanding the driver icon.



The expanded view of a top-level driver icon displays the following:

- The driver name
- The driver module (for example, com.novell.nds.dirxml.driver.delimitedtext.DelimitedTextDriver)

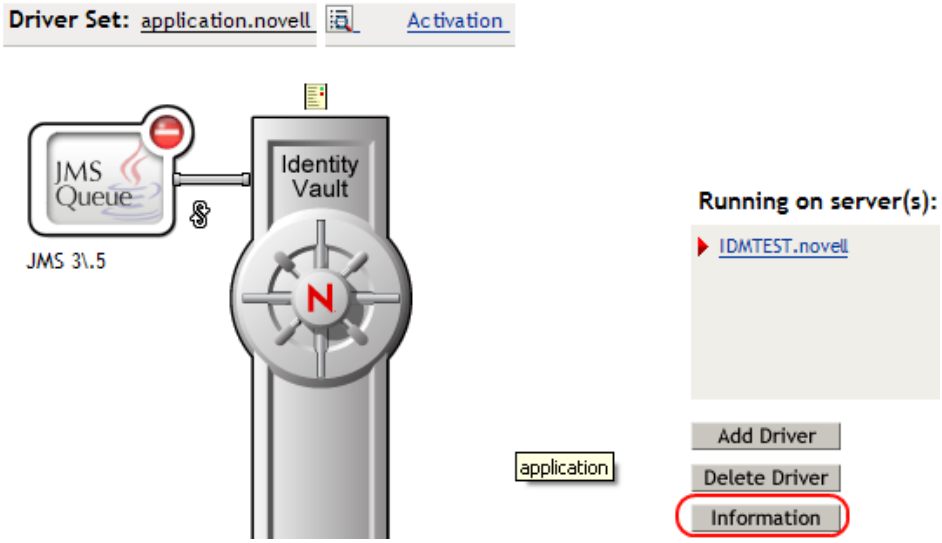
The expanded view of a server under a driver icon displays the following:

- The driver ID
- The version of the instance of the driver running on that server

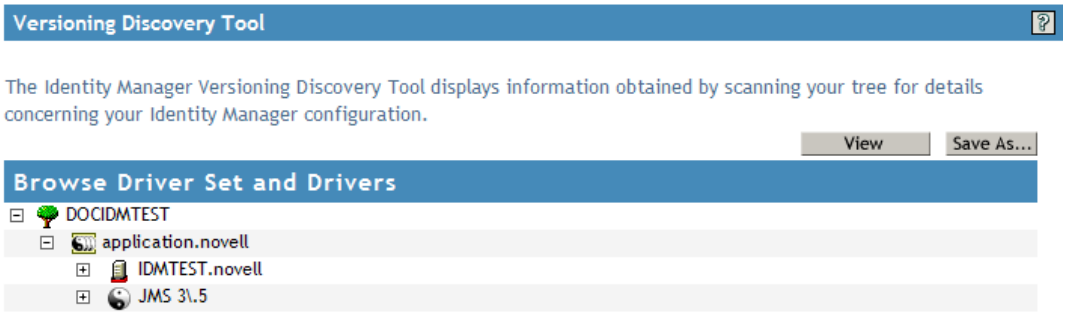
9.3.2 Viewing the Versioning Information As a Text File

Identity Manager publishes versioning information to a file. You can view this information in text format. The textual representation is the same information contained in the hierarchical view.

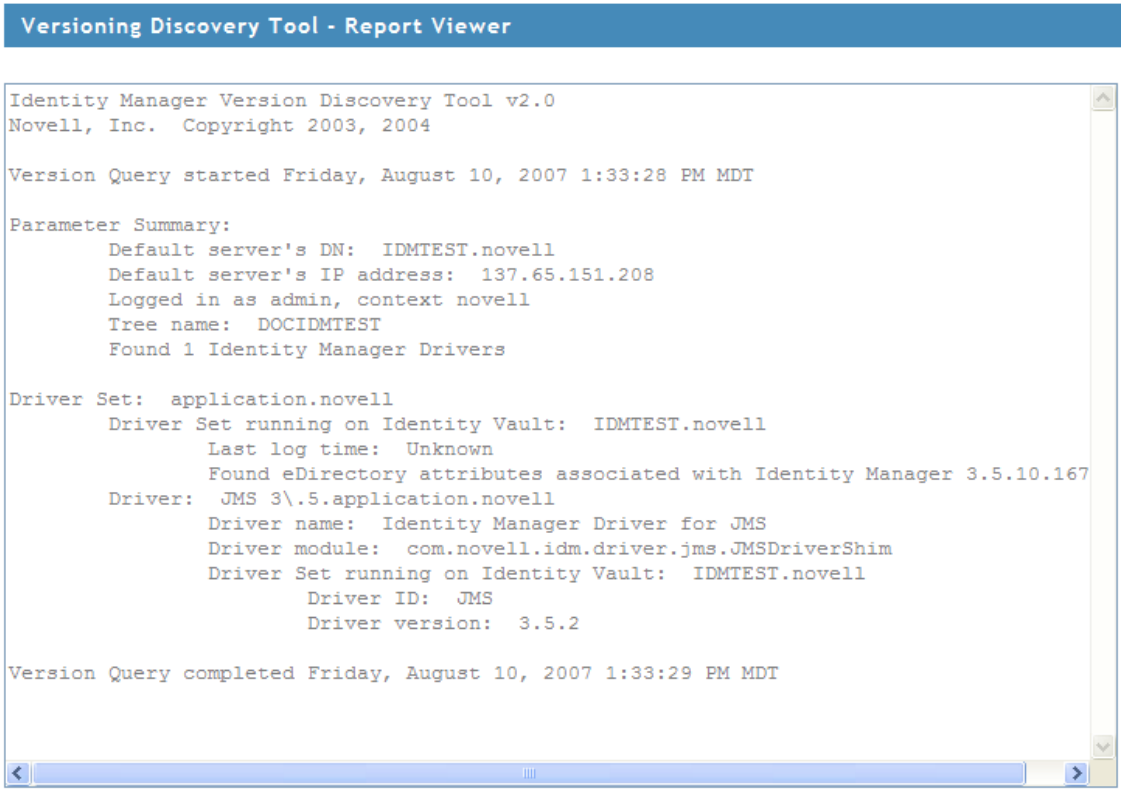
1. To find your Driver Set object in iManager, click *Identity Manager* > *Identity Manager Overview*, then click *Search*.
2. In the Identity Manager Overview, click *Information*.



   You can also select *Identity Manager Utilities* > *Versioning Discovery*, browse to and select the Driver Set object, then click *Information*.
3. In the Versioning Discovery Tool dialog box, click *View*.



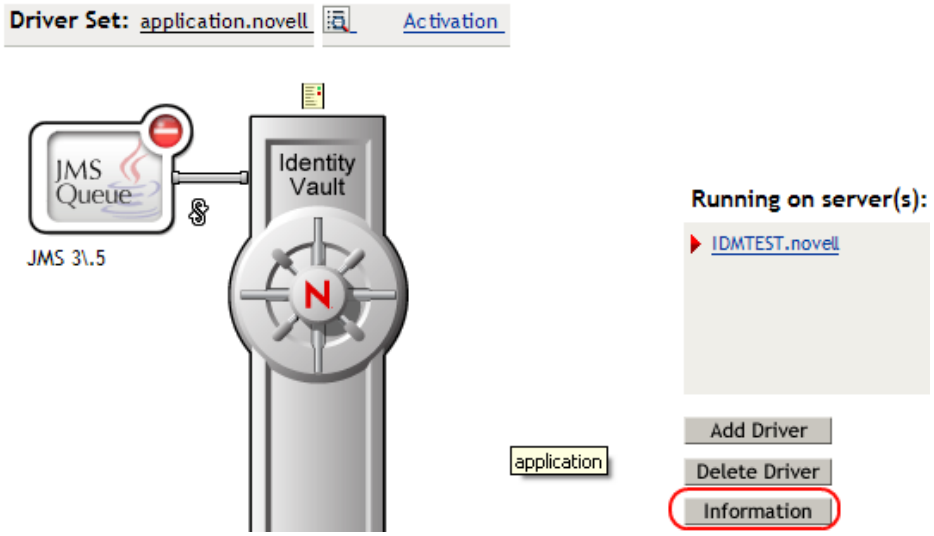
The information is displayed as a text file in the Report Viewer window.



9.3.3 Saving Versioning Information

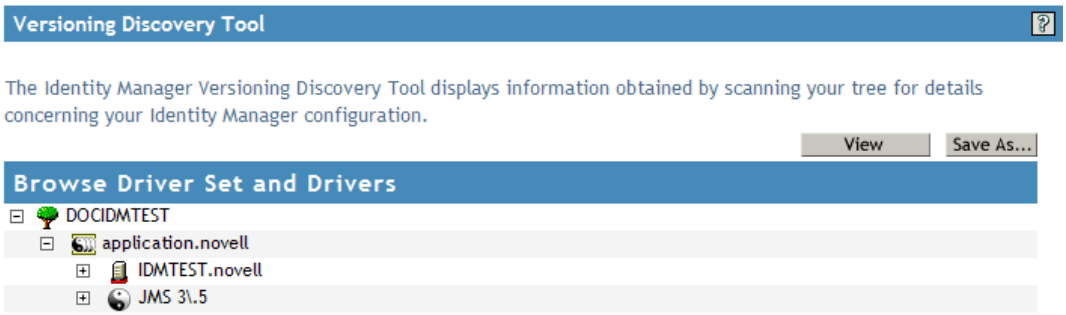
You can save versioning information to a text file on your local or network drive.

1. To find the Driver Set object in iManager, click *Identity Manager* > *Identity Manager Overview*, then click *Search*.
2. In the Identity Manager Overview, click *Information.*



You can also select *Identity Manager Utilities* > *Versioning Discovery*, browse to and select the Driver Set object, then click *Information*.

3 In the Versioning Discovery Tool dialog box, click *Save As*.



4 In the File Download dialog box, click *Save*.

5 Navigate to the desired directory, type a filename, then click *Save*.

Identity Manager saves the data to a text file.

9.4 Reassociating a Driver Object with a Server

The Driver object should always be associated with a server. If the driver is not associated with a server, the driver cannot start.

If the association becomes invalid, you see one of the following conditions:

- When upgrading eDirectory your Identity Manager server, you get the error UniqueSPIException error -783.
- No server is listed next to the driver in the Identity Manager Overview window.
- A server is listed next to the driver in the Identity Manager Overview window, but the name is garbled text.

To resolve this issue, disassociate the driver object and the server, then reassociate them.

1 In iManager click *Identity Manager* > *Identity Manager Overview*, then click *Search* to find the Driver Set object that the driver should be associated with.

2 Click the *Remove server* icon, then click *OK*.

3 Click the *Add server* icon, then browse to and select the server object.

4 Click *OK*.

9.5 Changing the Driver Configuration

If you need to change the driver configuration, Identity Manager allows you to make the change through iManager or Designer.

IMPORTANT: Do not run the driver wizard again, because it can overwrite information.

To change the driver configuration in iManager:

1. Click *Identity Manager > Identity Manager Overview*, then click *Search* to search for the driver set that is associated with the driver.
2. Browse to the driver, then click the upper right corner of the driver icon.
3. Click *Edit Properties*.

To change the driver configuration in Designer:

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

9.6 Storing Driver Passwords Securely with Named Passwords

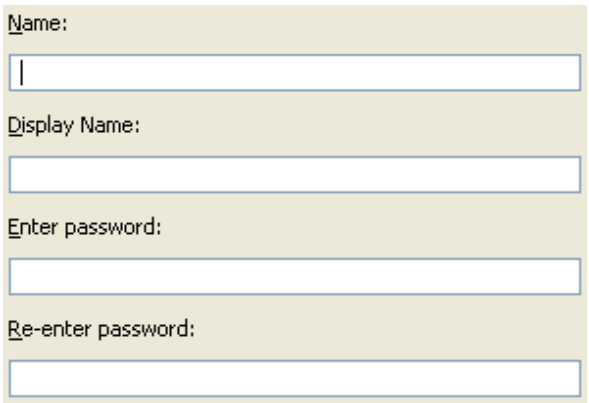
Identity Manager allows you to store multiple passwords securely for a particular driver. This functionality is referred to as Named Passwords. Each different password is accessed by a key, or name.

You can also use the Named Passwords feature to store other pieces of information securely, such as a user name.

To use a named password in a driver policy, you refer to it by the name of the password, instead of using the actual password, and the Metadirectory engine sends the password to the driver. The method described in this section for storing and retrieving Named Passwords can be used with any driver without making changes to the driver shim.

9.6.1 Using Designer to Configure Named Passwords

1. Right-click the Driver object, then select *Properties*.
2. Select *Named Password*, then click *New*.



3 Specify the *Name* of the Named Password.

4 Specify the *Display name* of the Named Password.

5 Specify the Named Password, then re-enter the password.

6 Click *OK* twice.

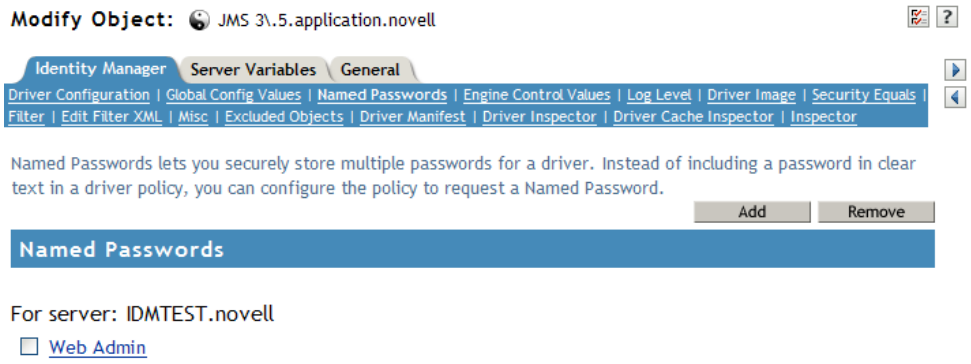
9.6.2 Using iManager to Configure Named Passwords

1 Click *Identity Manager* > *Identity Manager Overview*, then click *Search* to search for the driver set that is associated with the driver.

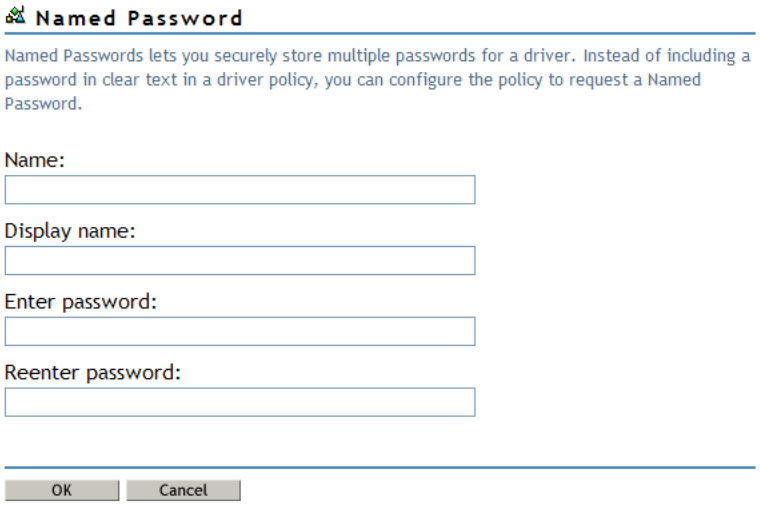
2 In the Identity Manager Overview, click the upper right corner of the driver icon, then click *Edit properties*.

3 On the Modify Object page on the Identity Manager tab, click *Named Passwords*.

The Named Passwords page appears, listing the current Named Passwords for this driver. If you have not set up any Named Passwords, the list is empty.



4 To add a Named Password, click *Add*, complete the fields, then click *OK*.



5 Specify a name, display name and a password, then click *OK* twice.

Keep in mind that you can use this feature to store other kinds of information securely, such as a username.

6 Click *OK* to restart the driver and have the changes take effect.

7 To remove a Named Password, select the password name, then click *Remove*.

The password is removed without prompting you to confirm the action.

9.6.3 Using Named Passwords in Driver Policies

- "Using the Policy Builder" on page 60
- "Using XSLT" on page 60

Using the Policy Builder

Policy Builder allows you to make a call to a named password. Create a new rule and select Named Password as the condition. You set an action depending upon if the named password is available or not available.

1 In Designer, launch Policy Builder, right-click, and click *New* > *Rule*.

2 Specify the name of the rule, then click *Next*.

3 Select the condition structure, then click *Next*.

4 Select *named password* for the *Condition*.

5 Browse to and select the named password that is stored on the driver.

In this example, it is *userinfo*.

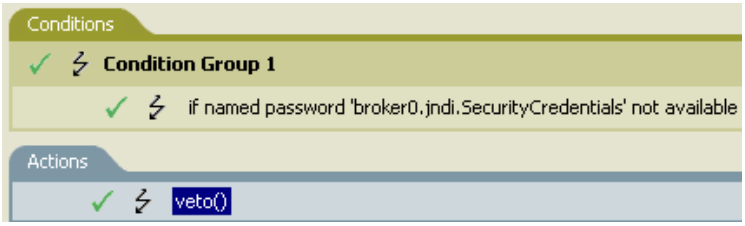
6 Select whether the Operator is available or not available.

7 Select an action for the *Do* field.

In this example, the action is *veto*.

The example indicates that if the userinfo named password is not available, then the event is vetoed.

Figure 9-1 *A Policy Using Named Password*



Using XSLT

The following example shows how a named password can be referenced in a driver policy on the Subscriber channel in XSLT:

```
<xsl:value-of
select="query:getNamedPassword($srcQueryProcessor,'mynamedpassword')"
xmlns:query="http://www.novell.com/java/
com.novell.nds.dirxml.driver.XdsQueryProcessor/>
```

9.6.4 Using the DirXML Command Line Utility to Configure Named Passwords

Creating a Named Password in the DirXML Command Line Utility

1. Run the DirXML Command Line utility.

   For information, see Appendix A, "DirXML Command Line Utility," on page 89.

2. Enter your username and password.

   The following list of options appears.

```
DirXML commands
 1: Start driver
 2: Stop driver
 3: Driver operations...
 4: Driver set operations...
 5: Log events operations...
 6: Get DirXML version
99: Quit
Enter choice:
```

3. Enter `3` for driver operations.

   A numbered list of drivers appears.

4. Enter the number for the driver you want to add a Named Password to.

   The following list of options appears.

```
Select a driver operation for:
driver_name
 1: Start driver
 2: Stop driver
 3: Get driver state
 4: Get driver start option
 5: Set driver start option
 6: Resync driver
 7: Migrate from application into DirXML
 8: Submit XDS command document to driver
 9: Check object password
10: Initialize new driver object
11: Passwords operations
12: Cache operations
99: Exit
Enter choice:
```

5. Enter `11` for password operations.

   The following list of options appears.

```
Select a password operation
 1: Set shim password
 2: Reset shim password
 3: Set named password
```

```
 4: Clear named password(s)
 5: List named passwords
99: Exit
Enter choice:
```

6 Enter `3` to set a new Named Password.

The following prompt appears:

```
Enter password name:
```

7 Enter the name by which you want to refer to the Named Password.

8 Enter the actual password that you want to secure at the following prompt:

```
Enter password:
```

The characters you type for the password are not displayed.

9 Confirm the password by entering it again at the following prompt:

```
Confirm password:
```

10 After you enter and confirm the password, you are returned to the password operations menu.

11 After completing this procedure, you can use the 99 option twice to exit the menu and quit the DirXML Command Line Utility.

Using the DirXML Command Line Utility to Remove a Named Password

This option is useful if you no longer need Named Passwords that you previously created.

1 Run the DirXML Command Line utility.

For information, see Appendix A, “DirXML Command Line Utility,” on page 89.

2 Enter your username and password.

The following list of options appears.

```
DirXML commands
 1: Start driver
 2: Stop driver
 3: Driver operations...
 4: Driver set operations...
 5: Log events operations...
 6: Get DirXML version
99: Quit
Enter choice:
```

3 Enter `3` for driver operations.

A numbered list of drivers appears.

4 Enter the number for the driver you want to remove Named Passwords from.

The following list of options appears.

```
Select a driver operation for:
driver_name
 1: Start driver
 2: Stop driver
 3: Get driver state
 4: Get driver start option
 5: Set driver start option
 6: Resync driver
```

```
 7: Migrate from application into DirXML
 8: Submit XDS command document to driver
 9: Check object password
10: Initialize new driver object
11: Passwords operations
12: Cache operations
99: Exit
Enter choice:
```

5 Enter `11` for password operations.

The following list of options appears.

```
Select a password operation
 1: Set shim password
 2: Reset shim password
 3: Set named password
 4: Clear named password(s)
 5: List named passwords
99: Exit
Enter choice:
```

6 (Optional) Enter `5` to see the list of existing Named Passwords.

The list of existing Named Passwords is displayed.

This step can help you make sure you are removing the correct password.

7 Enter `4` to remove one or more Named Passwords.

8 Enter `No` to remove a single Named Password at the following prompt:

```
Do you want to clear all named passwords? (yes/no):
```

9 Enter the name of the Named Password you want to remove at the following prompt:

```
Enter password name:
```

After you enter the name of the Named Password you want to remove, you are returned to the password operations menu:

```
Select a password operation
 1: Set shim password
 2: Reset shim password
 3: Set named password
 4: Clear named password(s)
 5: List named passwords
99: Exit
Enter choice:
```

10 (Optional) Enter `5` to see the list of existing Named Passwords.

This step lets you verify that you have removed the correct password.

11 After completing this procedure, you can use the 99 option twice to exit the menu and quit the DirXML Command Line utility.

9.7 Adding Driver Heartbeat

The driver heartbeat is a feature of the Identity Manager drivers that ship with Identity Manager 2 and later. Its use is optional. Driver heartbeat is configured by using a driver parameter with a time interval specified. If a heartbeat parameter exists and has an interval value other than 0, the driver

sends a heartbeat document to the Metadirectory engine if there is no communication on the Publisher channel for the specified interval of time.

The intent of the driver heartbeat is to give you a trigger to allow you to initiate an action at regular intervals, if the driver does not communicate on the Publisher channel as often as you want the action to occur. To take advantage of the heartbeat, you must customize your driver configuration or other tools. The Metadirectory engine accepts the heartbeat document but does not take any action because of it.

For most drivers, a driver parameter for heartbeat is not used in the sample configurations, but you can add it.

A custom driver that is not provided with Identity Manager can also provide a heartbeat document, if the driver developer has written the driver to support it.

To configure the heartbeat:

1. In iManager, click *Identity Manager* > *Identity Manager Overview*.
2. Browse to and select your Driver Set object, then click *Search*.
3. In Identity Manager Overview, click the upper right corner of the driver icon, then click *Edit properties*.
4. On the Identity Manager tab, click *Driver Configuration*, scroll to Driver Parameters, then look for Heart Beat or a similar display name.

   If a driver parameter already exists for heartbeat, you can change the interval and save the changes, and configuration is then complete.

   The value of the interval cannot be less than 1. A value of 0 means the feature is turned off.

   The unit of time is usually minutes; however, some drivers might choose to implement it differently, such as using seconds.
5. If a driver parameter does not exist for heartbeat, click *Edit XML*.
6. Add a driver parameter entry like the following example, as a child of `<publisher-options>`.

   ```
   <pub-heartbeat-interval display-name="Heart Beat">10</pub-heartbeat-interval>
   ```

   TIP: If the driver does not produce a heartbeat document after being restarted, check the placement of the driver parameter in the XML.
7. Save the changes, and make sure the driver is stopped and restarted.

After you have added the driver parameter, you can edit the time interval by using the graphical view. Another option is to create a reference to a global configuration value (GCV) for the time interval. Like other global configuration values, the driver heartbeat can be set at the Driver Set level instead of on each individual Driver object. If a driver does not have a particular global configuration value, and the Driver Set object does have it, the driver inherits the value from the Driver Set object.

Customizing the Driver

10

Identity Manager contains a default driver template that is configured for basic tasks. Most people find that they need to customize the driver to work with their business needs. Identity Manager is designed so that you can customize the driver to work in your environment.

In order to customize the driver, you need to know what the default template does. The following sections explain what the default functionality of the driver is and how to customize the driver to meet your business needs.

10.1 Default Driver Parameters

The driver import file defines some basic functionality in the driver. How the driver handles this information for each channel is explained in the tables below.

Table 10-1 *Driver Settings*

Parameter	Sub-Parameter	Description
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.
Show connection-related parameters		Displays connection-related parameters such as JNDI connection factory names and usernames or passwords.
	Username	Specifies the username used to authenticate to the message broker.
	Password	The password used to authenticate to the message broker. After entering the password, you need to re-enter it for validation.
	Show queue connection factory options	Specifies the JNDI name of the connection factory used to create connections to queues.

Parameter	Sub-Parameter	Description
	Show topic connection factory options	1. Specify the JNDI name of the connection factory used to create connections to topics. 2. Specify the Client ID used to create 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 parameters		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.
	INITIAL_CONTEXT_ FACTORY	The name that uniquely identifies this JNDI context property.
	Value	The name of the Java class used to create a JNDI context for this message broker.
	PROVIDER_URL	The name that uniquely identifies this JNDI context property
	Value	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
	SECURITY_ CREDENTIALS	The name that uniquely identifies this JNDI context property.
	Value	The password used to authenticate to this message broker.
	SECURITY_ PRINCIPAL	The name that uniquely identifies this JNDI context property.
	Value	The username used to authenticate to this message broker.
	URL_PKG_PREFIXES	The name that uniquely identifies this JNDI context property.
	Value	The value of this JNDI context property.
	Show remaining standard properties	Displays remaining, less commonly used standard JNDI context properties
	APPLET	The name that uniquely identifies this JNDI context property.
	Value	The value of this JNDI context property.

Parameter	Sub-Parameter	Description
	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		Displays vendor-specific JNDI context properties.
	Name	The name that uniquely identifies this JNDI context property.
	Value	The value of this JNDI context property.

Table 10-2 *Subscriber Channel*

Parameter	Sub-Parameter	Description
Disable subscriber		Select *yes* to prevent this channel from sending messages to JMS providers.

Parameter	Sub-Parameter	Description
Show default message options		Displays options that are global to all messages.
	Default message expiration (milliseconds)	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.
	Default message priority	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.
	Default message type	Specifies the default message type as text or bytes. This setting is global for all sent messages.
	Show default message properties	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.
	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: 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.
	Value	Message property value.
Show default destination options		Displays options global to all destinations.
	Default destination type	Specifies whether all destinations are queues or topics by default. This setting is global for all destinations.

Parameter	Sub-Parameter	Description
	Default omit message envelope	Specifies whether the JMS message envelope should be omitted from received messages. This setting is global to all destinations.
	Default receive timeout (seconds)	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. This setting is global to all destinations.
	Default message filter	Specifies how destinations filter received messages. This setting is global to all destinations.
	Default message selector	Specifies 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'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.
Destination unique id		Specifies the identifier for this destination by which it is known in the IDM namespace. This name is also the durable subscription name for topics. This value must be unique per channel (Subscriber/ Publisher).
Show additional destination options		Displays additional options for this selected destination.
	Destination JNDI name	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.
	Destination type	Specifies whether this destination is a queue or a topic.
	Destination mode	Specifies whether the destination is used to send or receive messages.
	Message type	Specifies whether messages are sent as text or bytes.

Parameter	Sub-Parameter	Description
	Show message properties	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.
	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: 1. Standard JMS properties. They usually begin with “JMS” or “JMSX”. 2. Provider-specific properties. They begin with “JMS_”. 3. Application-specific. Anything else.
	Value	Message property value.
Destination unique id		Specifies the identifier by which this destination is known in the IDM namespace. This name is also the durable subscription name for topics. This value must be unique per channel (Subscriber/ Publisher).
Show additional destination options		Displays additional options for this selected destination.
	Destination JNDI name	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.
	Destination type	Specifies whether the destination is a queue or a topic.
	Destination mode	Specifies whether the destination is used to send or receive messages.
	Message type	Specifies whether messages should be sent as text or bytes.
	Show message properties	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.

Parameter	Sub-Parameter	Description
	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: 1. Standard JMS properties. They usually begin with “JMS” or “JMSX”. 2. Provider-specific properties. They begin with “JMS_”. 3. Application-specific. Anything else.
	Value	Message property value.
	Omit message envelope	Specifies whether the JMS message envelope is omitted from messages received by this destination.
	Receive timeout (seconds)	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.
	Message filter	Specifies how the destination receives filtered messages.
	Message selector	Specifies a custom message selector to filter received messages. Message selectors are like SQL WHERE clauses (for example, 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.
Destination unique id		Specifies the identifier by which this destination is known in the IDM namespace. This name is also the durable subscription name for topics. This value must be unique per channel (Subscriber/ Publisher).
Show additional destination options		Displays additional options for this destination.
	Destination JNDI name	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.
	Destination type	Specifies whether the destination is a queue or a topic.
	Destination mode	Specifies whether the destination is used to send or receive messages.

Parameter	Sub-Parameter	Description
	Message type	Specifies whether messages should be sent as text or bytes.
	Show message properties	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.
	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: 1. Standard JMS properties. They usually begin with “JMS” or “JMSX”. 2. Provider-specific properties. They begin with “JMS_”. 3. Application-specific. Anything else.
	Value	Message property value.

Table 10-3 *Publisher Channel*

Parameter	Sub-parameter	Description
Disable publisher		Select *yes* to prevent this channel from receiving messages from JMS providers.
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		Displays options global to all messages.
	Default message expiration (milliseconds)	Specifies how long messages live after they reach a destination. Specify the time duration in milliseconds. 0 means the messages live indefinitely. This setting is global for all sent messages.

Parameter	Sub-parameter	Description
	Default message priority	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.
	Default message type	Specifies whether messages are text or byte. This setting is global for all sent messages.
	Show default message properties	Displays the parameters that specify 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	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 begin with “JMS_”. 3. Application specific. Anything else.
	Value	Message property value.
Show default session options		Displays options that are global to all sessions.
	Default message acknowledgment threshold	Specifies how many messages are received by a monitored destination before an acknowledgment is sent to the broker.
Show default destination options		Displays options that are global to all destinations.
	Default destination type	Specifies whether destinations are topics or queues by default. This setting is global for all destinations.

Parameter	Sub-parameter	Description
	Default omit message envelope	Specifies if the JMS message envelope is omitted from received messages. This setting is global for all destinations.
	Default receive timeout (seconds)	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. This setting is global for all destinations.
	Default message filter	Specifies how the destination's filter receives messages. This setting is global for all destinations.
	Default message selector	Specifies 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'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.
	Default polling interval (milliseconds)	Specifies how often destinations are polled for new messages (in milliseconds.) This setting is global for all destinations.
Destination unique id		Specifies the identifier by which this destination is known in the IDM namespace. This name is also the durable subscription name for topics. This value must be unique per channel (Subscriber/Publisher).
Show additional destination options		Displays parameters for this selected destination.
	Destination JNDI name	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.
	Destination type	Specifies whether this destination is a queue or topic.
	Destination mode	Specifies whether this destination sends or receives messages.

Parameter	Sub-parameter	Description
	Message type	Specifies whether messages are sent in text or byte format.
	Show message properties	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.
	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: 1. Standard JMS properties. They usually begin with “JMS” or “JMSX”. 2. Provider-specific properties. They begin with “JMS_”. 3. Application-specific. Anything else.
	Value	Message property value.
Destination unique id		Specifies the identifier by which this destination is known in the IDM namespace. This name is also the durable subscription name for topics. This value must be unique per channel (Subscriber/Publisher).
Show additional destination options		Displays parameters for this selected destination.
	Destination JNDI name	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.
	Destination type	Specifies whether the destination is a queue or a topic.
	Destination mode	Specifies whether this destination sends or receives messages.
	Omit message envelope	Whether the JMS message envelope be omitted from messages received by this destination.
	Receive timeout (seconds)	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.

Parameter	Sub-parameter	Description
	Message filter	Specifies how this destination filters messages.
	Message selector	Specifies a custom message selector to filter received messages. Message selectors are like SQL WHERE clauses For example, 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.
	Is durable	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.
	Subscription name	Specify the name of the durable subscription to create on the broker. **NOTE:** This resource might need to be cleaned up manually when this driver is deleted unless specific procedures are followed.
	Actively monitor	Specify if you want the channel to periodically monitor this destination for messages.

10.1.1 Showing Additional Destinations

The default configuration includes the ability to show two destinations each for the Publisher and Subscriber channels.

To add more destinations:

1. In iManager, on the Driver Parameter's page, click *Edit XML*.
2. Look for a `<group>` element. Within the group, look for an element that begins with `<definition display-name="Show destination". . .>`
3. Copy the entire contents of the `<group>` element and paste it after the other entries for `Show destination`.
4. Within the `Show destination` element, there's an entry for `"name=singorex"` or `"name=pignorex"` (signore is for Subscriber channel destinations; pignore is for Publisher channel destinations). You should look for the highest number and increment the number.
5. Scroll down through the copied XML and locate the `<definition display-name="Destination unique id" gcv-ref="destination_1.unique-id">` element. You also need to increment the destination_x number to be the same number as the signore, pignore value.

6 Save your XML.

You can repeat this process as many times as needed to add additional destinations for the Publisher or Subscriber channel.

10.1.2 Considerations When Removing the Driver

If you decide to delete the driver for a Publish/Subscribe configuration that is using a durable subscription, there are several steps you must manually complete to ensure that the subscription is removed.

1 In iManager, edit the driver object. For every destination, you must delete the Destination Unique ID and Destination JNDI Name.

2 Restart the driver.

3 Open the driver object, then click *General* > *Other*.

4 Select the DirXML-DriverStorage attribute, then click *Edit*.

5 Confirm that there are no child elements of <subscription/>. If none exist, the durable subscription has been removed from the message system.

10.2 Customizing the Driver

10.2.1 Customizing Policies

To change the default functionality of the driver, use Policy Builder to change the policies. For more information, see the *Understanding Policies for Identity Manager 3.5.1*.

10.2.2 Customizing Driver Parameters

You can also change the default functionality of the driver by changing the driver’s parameters.

1 In iManager, click *Identity Manager* > *Identity Manager Overview*.

2 Browse to the Driver Set where the driver exists, then click *Search*.

3 Click the upper right corner of the driver icon and select *Edit properties*.

4 Select *Driver Configuration* and scroll down to *Driver Parameters*.

5 Make the changes you want, then click *OK*.

Security: Best Practices

11

In order to secure the driver and the information it is synchronizing, see “Security: Best Practices” in the *Novell Identity Manager 3.5.1 Administration Guide*.

Backing Up the Driver

12

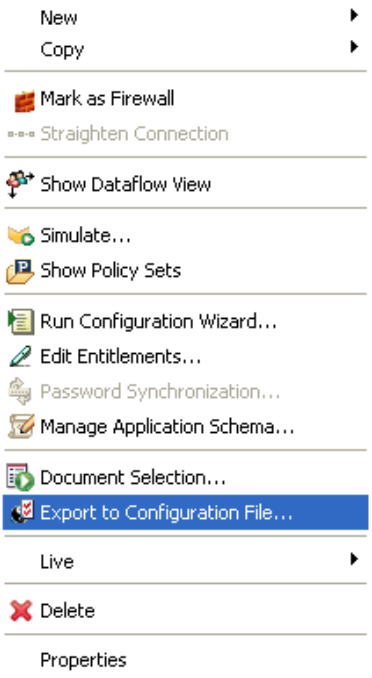
You can use Designer or iManager to create an XML file of the driver. The file contains all of the information entered into the driver during configuration. If the driver becomes corrupted, the exported file can be imported to restore the configuration information.

IMPORTANT: If the driver has been deleted, all of the associations on the objects are deleted. When the XML file is imported, all of the associations are re-created through the migration process.

Not all server-specific information stored on the driver is contained in the XML file. Make sure this information is documented through the Doc Gen process in Designer.

12.1 Exporting the Driver in Designer

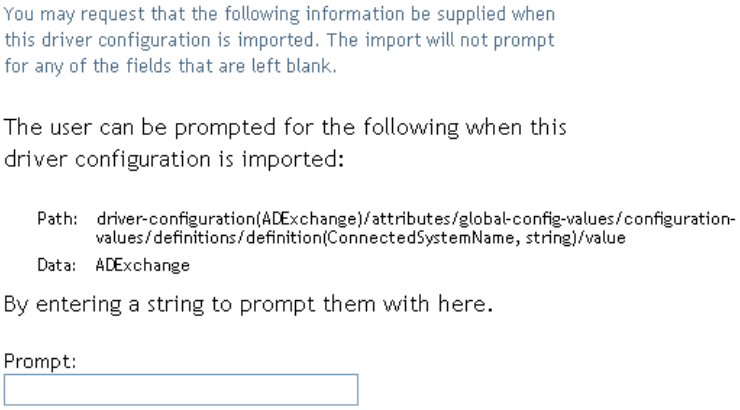
1. Open a project in Designer, then right-click the driver object.
2. Select *Export Driver to Configuration File*.



3. Specify a unique name for the configuration file, browse to location where it should be saved, then click *Save*.
4. Click *OK* in the Export Configuration Results window.

12.2 Exporting the Driver in iManager

1. In iManager, select *Identity Manager* > *Identity Manager Overview*.
2. Browse to and select the Driver Set object, then click *Search*.
3. Click the driver icon.
4. Select *Export* on the Identity Manager Driver Overview window.
5. Browse to and select the Driver object you want to export, then click *Next*.
6. Select *Export all policies, linked to the configuration or not* or select *Only export policies that are linked to the configuration*, depending upon the information you want to have stored in the XML file.
7. Click *Next*.
8. If you want prompts for information to appear when the driver is imported, fill in the appropriate field; otherwise, click *Next*.



9. Click *Save As*, then click *Save*.
10. Browse and select a location to save the XML file, then click *Save*.
11. Click *Finish*.

Troubleshooting

13

Viewing driver processes is necessary to analyze unexpected behavior.

13.1 Viewing Driver Processes

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.

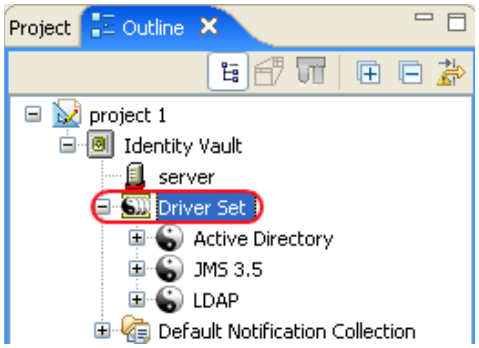
In order to see the driver processes in DSTRACE, values are added to the Driver Set and the driver objects. You can do this in Designer and iManager.

13.1.1 Adding Trace Levels in Designer

You can add trace levels to the Driver Set object or to each Driver object.

Driver Set

1 In an open project in Designer, select the Driver Set object in the Outline view.



2 Right-click and select *Properties*, then click *5. Trace*.

3 Set the parameters for tracing, then click *OK*.

See Table 13-1 on page 84 for more information about the Driver Set trace parameters.

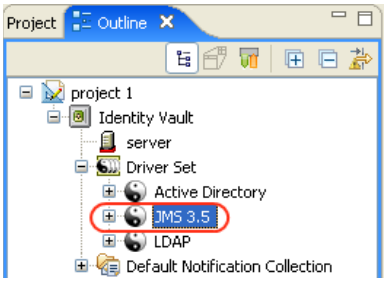
If you set the trace level on the Driver Set object, all drivers appear in the DSTRACE logs.

Table 13-1 *Driver Set Trace Parameters*

Parameter	Description
Driver trace level	As the Driver object trace level increases, the amount of information displayed in DSTRACE increases. Trace level 1 shows errors, but not the cause of the errors. If you want to see password synchronization information, set the trace level to 5.
XSL trace level	DSTRACE displays XSL events. Set this trace level only when troubleshooting XSL style sheets. If you do not want to see XSL information, set the level to zero.
Java debug port	Allows developers to attach a Java debugger.
Java trace file	When a value is set in this field, all Java information for the Driver Set object is written to a file. The value for this field is the path for that file. As long as the file is specified, Java information is written to this file. If you do not need to debug Java, leave this field blank.
Trace file size limit	Allows you to set a limit for the Java trace file. If you set the file size to *Unlimited*, the file grows in size until there is no disk space left. **NOTE:** The trace file is created in multiple files.Identity Manager automatically divides the maximum file size by ten and creates ten separate files. The combined size of these files equals the maximum trace file size.

Driver

1 In an open project in Designer, select the Driver object in the Outline view.



2 Right-click and select *Properties*, then click *8. Trace*.

3 Set the parameters for tracing, then click *OK*.

See Table 13-2 for more information about these parameters.

If you set the parameters only on the Driver object, only information for that driver appears in the DSTRACE log.

Table 13-2 *Driver Trace Parameters*

Parameter	Description
Trace level	As the Driver object trace level increases, the amount of information displayed in DSTRACE increases. Trace level 1 shows errors, but not the cause of the errors. If you want to see password synchronization information, set the trace level to 5. if you select *Use setting from Driver Set*, the value is taken from the Driver Set object.
Trace file	Specify a filename and location for where the Identity Manager information is written for the selected driver. if you select *Use setting from Driver Set*, the value is taken from the Driver Set object.
Trace file size limit	Allows you to set a limit for the Java trace file. If you set the file size to *Unlimited*, the file grows in size until there is no disk space left. if you select *Use setting from Driver Set*, the value is taken from the Driver Set object. **NOTE:** The trace file is created in multiple files.Identity Manager automatically divides the maximum file size by ten and creates ten separate files. The combined size of these files equals the maximum trace file size.
Trace name	The driver trace messages are prepended with the value entered instead of the driver name. Use this option if the driver name is very long.

13.1.2 Adding Trace Levels in iManager

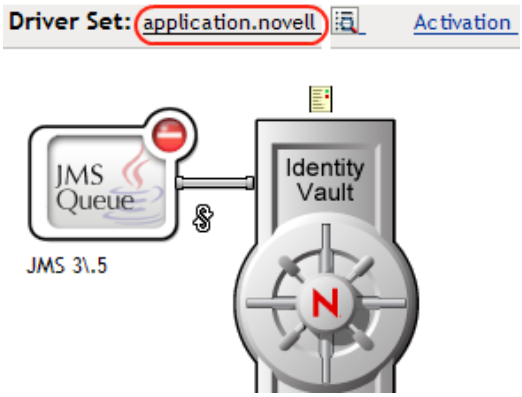
You can add trace levels to the Driver Set object or to each Driver object.

- “Driver Set” on page 85
- “Driver” on page 86

Driver Set

1. In iManager, select *Identity Manager* > *Identity Manager Overview*.
2. Browse to the Driver Set object, then click *Search*.

3 Click the Driver Set name.



4 Select the *Misc* tab for the Driver Set object.

5 Set the parameters for tracing, then click *OK*.

See Table 13-1 on page 84 for more information about these parameters.

Driver

1 In iManager, select *Identity Manager* > *Identity Manager Overview*.

2 Browse to the Driver Set object where the Driver object resides, then click *Search*.

3 Click the upper right corner of the Driver object, then click *Edit properties*.

4 Select the *Misc* tab for the Driver object.

5 Set the parameters for tracing, then click *OK*.

See Table 13-2 on page 85 for more information.

NOTE: The option *Use setting from Driver Set* does not exist in iManager.

13.1.3 Capturing Driver Processes to a File

You can save driver processes to a file by using the parameter on the Driver object or by using DSTRACE. The parameter on the Driver object is the *Trace file* parameter, under the *MISC* tab.

The following methods helps you capture and save Identity Manager processes through DSTRACE on different platforms.

- “NetWare” on page 87
- “Windows” on page 87
- “UNIX” on page 87
- “iMonitor” on page 88

NetWare

Use `dstrace.nlm` to display trace messages on the system console or trace messages to a file (`sys:\system\dstrace.log`). `dstrace.nlm` displays the trace messages to a screen labeled DSTrace Console.

1. Enter `dstrace.nlm` at the server console to load `dstrace.nlm` into memory.
2. Enter `dstrace screen on` at the server console to allow trace messages to appear on the `DSTrace Console` screen.
3. Enter `dstrace file on` at the server console. to cpature trace messages sent to the DSTrace Console to the `dstrace.log`.
4. (Optional) Enter `dstrace -all` at the server console to make it easier to read the trace log.
5. Enter `dstrace +dxml dstrace +dvrs` at the server console to display Identity Manager events.
6. Enter `dstrace +tags dstrace +time` at the server console to display message tags and time stamps.
7. Toggle to the DSTrace Console screen and watch for the event to pass.
8. Toggle back to the server console.
9. Type `dstrace file off` at the server console.

   This stops capture trace messages to the log file. It also stops logging the information into the file.
10. Open the `dstrace.log` in a text editor and search for the event or the object you modified.

Windows

1. Open the *Control Panel* > *NDS Services* > `dstrace.dlm`, then click *Start* to display the NDS Server Trace utility window.
2. Click *Edit* > *Options*, then click *Clear All* to clear all of the default flags.
3. Select *DirXML* and *DirXML Drivers*.
4. Click OK.
5. Click *File* > *New*.
6. Specify the filename and location where you want the DSTRACE information saved, then click *Open*.
7. Wait for the event to occur.
8. Click *File* > *Close*.

   This stops the information from being written to the log file.
9. Open the file in a text editor and search for the event or the object you modified.

UNIX

1. Enter `ndstrace` to start the ndstrace utility.
2. Enter `set ndstrace=nodebug` to turn off all trace flags currently set.
3. Enter `set ndstrace on` to display trace messages to the console.
4. Enter `set ndstrace file on` to capture trace messages to the file `ndstrace.log` to the directory where eDirectory is installed. By default it is `/var/nds`.

5. Enter `set ndstrace=+dxml` to display the Identity Manager events.
6. Enter `set ndstrace=+dvrs` to display the Identity Manager driver events.
7. Wait for the event to occur.
8. Enter `set ndstrace file off` to stop the logging of information to file.
9. Enter `exit` to quite the ndstrace utility.
10. Open the file in a text editor. Search for the event or the object that was modified.

iMonitor

iMonitor allows you to get DSTRACE information from a Web browser. It does not matter where Identity Manager is running. The following files run iMonitor:

- `ndsimon.nlm` runs on NetWare®.
- `ndsimon.dlm` runs on Windows.
- `ndsimonitor` runs on UNIX.

1. Access iMonitor from http://*server_ip*:8008/nds.

   Port 8008 is the default.
2. Specify a username and password with administrative rights, then click *Login*.
3. Select *Trace Configuration* on the left side.
4. Click *Clear All*.
5. Select *DirXML* and *DirXML Drivers*.
6. Click *Trace On*.
7. Select *Trace History* on the left side.
8. Click the document with the Modification Time of Current to see a live trace.
9. Change the *Refresh Interval* if you want to see information more often.
10. Select *Trace Configuration* on the left side, then click *Trace Off* to turn the tracing off.
11. Select *Trace History* to view the trace history.

    The files are distinguished by their time stamp.

If you need a copy of the HTML file, the default location is:

- NetWare: `sys:\system\ndsimon\dstrace*.htm`
- Windows: `Drive_letter:\novell\nds\ndsimon\dstrace\*.htm`
- UNIX: `/var/nds/dstrace/*.htm`

DirXML Command Line Utility

A

The DirXML® Command Line utility allows you to use a command line interface to manage the driver. You can create scripts to manage the driver with the commands.

The utility and scripts are installed on all platforms during the Identity Manager installation. The utility is installed to the following locations:

- Windows: `\Novell\Nds\dxcmd.bat`
- NetWare®: `sys:\system\dxcmd.ncf`
- UNIX: `/usr/bin/dxcmd`

There are two different methods for using the DirXML Command Line utility:

- Section A.1, “Interactive Mode,” on page 89
- Section A.2, “Command Line Mode,” on page 98

A.1 Interactive Mode

The interactive mode provides a text interface to control and use the DirXML Command Line utility.

1. At the console, enter `dxcmd`.
2. Enter the name of a user with sufficient rights to the Identity Manager objects, such as admin.novell.
3. Enter the user’s password.

4. Enter the number of the command you want to perform.

   Table A-1 on page 90 contains the list of options and what functionality is available.
5. Enter 99 to quit the utility.

NOTE: If you are running eDirectory™ 8.8 on UNIX or Linux*, you must specify the -host and -port parameters. For example, `dxcmd -host 10.0.0.1 -port 524`. If the parameters are not specified, a jclient error occurs.

```
novell.jclient.JCException: connect (to address) 111 UNKNOWN ERROR
```

By default, eDirectory 8.8 is not listening to localhost. The DirXML Command Line utility needs to resolve the server IP address or hostname and the port to be able to authenticate.

Table A-1 *Interactive Mode Options*

Option	Description
1: *Start Driver*	Starts the driver. If there is more than one driver, each driver is listed with a number. Enter the number of the driver to start the driver.
2: *Stop Driver*	Stops the driver. If there is more than one driver, each driver is listed with a number. Enter the number of the driver to stop the driver.
3: *Driver operations*	Lists the operations available for the driver. If there is more than one driver, each driver is listed with a number. Enter the number of the driver to see the operations available. See Table A-2 on page 91 for a list of operations.
4: *Driver set operations*	Lists the operations available for the driver set. ◆ 1: Associate driver set with server ◆ 2: Disassociate driver set from server ◆ 99: Exit
5: *Log events operations*	Lists the operations available for logging events through Novell® Audit. See Table A-5 on page 95 for a description of these options.
6: *Get DirXML version*	Lists the version of the Identity Manager installed.
7: *Job operations*	Manages jobs created for Identity Manager.
99: *Quit*	Exits the DirXML Command Line utility

Figure A-1 *Driver Options*

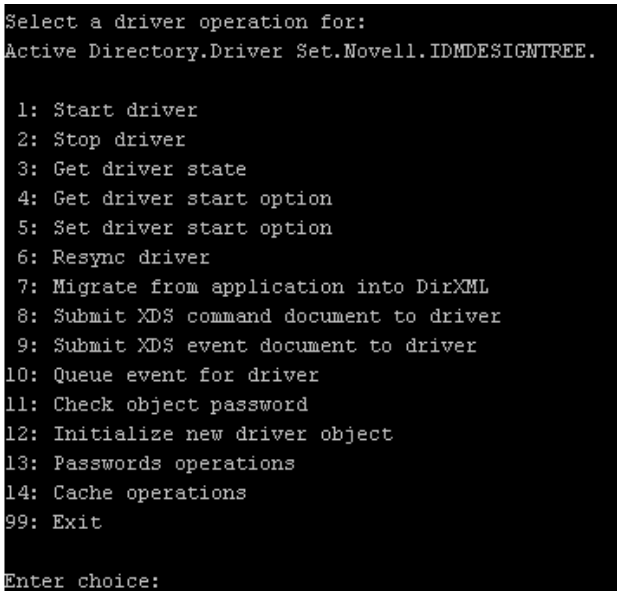


Table A-2 *Driver Options*

Options	Description
1: *Start driver*	Starts the driver.
2: *Stop driver*	Stops the driver.
3: *Get driver state*	Lists the state of the driver. ◆ 0 - Driver is stopped ◆ 1 - Driver is starting ◆ 2 - Driver is running ◆ 3 - Driver is stopping
4: *Get driver start option*	Lists the current driver start option. ◆ 1 - Disabled ◆ 2 - Manual ◆ 3 - Auto
5: *Set driver start option*	Changes the start option of the driver. ◆ 1 - Disabled ◆ 2 - Manual ◆ 3 - Auto ◆ 99 - Exit
6: *Resync driver*	Forces a resynchronization of the driver. It prompts for a time delay: *Do you want to specify a minimum time for resync? (yes/no)*. If you enter Yes, specify the date and time you want the resynchronization to occur: *Enter a date/time (format 9/27/05 3:27 PM)*. If you enter No, the resynchronization occurs immediately.
7: *Migrate from application into DirXML*	Processes an XML document that contains a query command: *Enter filename of XDS query document:* Create the XML document that contains a query command by using the Novell `nds.dtd` (http://developer.novell.com/ndk/doc/dirxml/dirxmlbk/ref/ndsdtd/query.html). Examples: NetWare: `sys:\files\query.xml` Windows: `c:\files\query.xml` Linux: `/files/query.xml`

Options	Description
8: *Submit XDS command document to driver*	Processes an XDS command document: *Enter filename of XDS command document:* Examples: NetWare: `sys:\files\user.xml` Windows: `c:\files\user.xml` Linux: `/files/user.xml` *Enter name of file for response:* Examples: NetWare: `sys:\files\user.log` Windows: `c:\files\user.log` Linux: `/files/user.log`
9: Submit XDS event document to driver	Processes an XDS event document: *Enter filename of XDS event document:* Examples: NetWare: `sys:\files\add.xml` Windows: `c:\files\add.xml` Linux: `/files/add.xml`
10: Queue event for driver	Adds and event to the driver queue *Enter filename of XDS event document:* Examples: NetWare: `sys:\files\add.xml` Windows: `c:\files\add.xml` Linux: `/files/add.xml`
11: *Check object password*	Validates that an object's password in the connected system is associated with a driver. It matches the object's eDirectory password (Distribution Password, used with Universal Password). *Enter user name*:
12: *Initialize new driver object*	Performs an internal initialization of data on a new Driver object. This is only for testing purposes.
13: *Password operations*	There are nine Password options. See Table A-3 on page 93 for a description of these options.
14: *Cache operations*	There are five Cache operations. See Table A-4 on page 94 for a descriptions of these options.

Options	Description
99: *Exit*	Exits the driver options.

Figure A-2 *Password Operations*

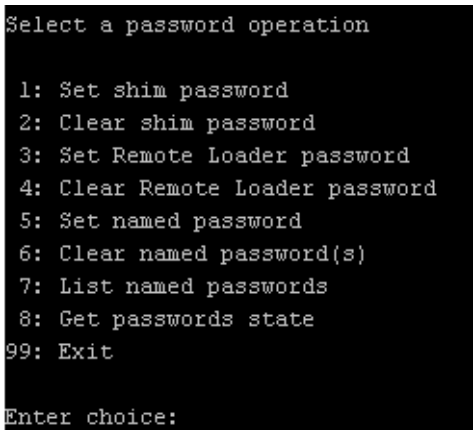


Table A-3 *Password Operations*

Operation	Description
1: *Set shim password*	Sets the application password. This is the password of the user account you are using to authenticate into the connected system with.
2: *Clear shim password*	Clears the application password.
3: *Set Remote Loader password*	The Remote Loader password is used to control access to the Remote Loader instance. Enter the Remote Loader password, then confirm the password by typing it again.
4: *Clear Remote Loader password*	Clears the Remote Loader password so no Remote Loader password is set on the Driver object.
5: *Set named password*	Allows you to store a password or other pieces of security information on the driver. There are four prompts to fill in: ◆ *Enter password name:* ◆ *Enter password description:* ◆ *Enter password:* ◆ *Confirm password:*
6: *Clear named passwords*	Clears a specified named password or all named passwords that are stored on the driver object: *Do you want to clear all named passwords? (yes/no).* If you enter Yes, all Named Passwords are cleared. If you enter No, you are prompted to specify the password name that you want to clear.

Operation	Description
7: *List named passwords*	Lists all named passwords that are stored on the driver object. It lists the password name and the password description.
8: *Get password state*	Lists if a password is set for: ◆ Driver Object password ◆ Application password ◆ Remote loader password The dxcmd utility allows you to set the Application password and the Remote Loader password. You cannot set the Driver Object password with this utility. It shows if the password has been set or not.
99: *Exit*	Exits the current menu and takes you back to the Driver options.

Figure A-3 *Cache Operations*

```
Enter choice: 14


Select a cache operation

 1: Get driver cache limit
 2: Set driver cache limit
 3: View cached transactions
 4: Delete cached transactions
99: Exit

Enter choice:
```

Table A-4 *Cache Operations*

Operation	Description
1: *Get driver cache limit*	Displays the current cache limit that is set for the driver.
2: *Set driver cache limit*	Sets the driver cache limit in kilobytes. A value of 0 is unlimited.
3: *View cached transactions*	A text file is created with the events that are stored in cache. You can select the number of transactions to view. ◆ *Enter option token* (default=0): ◆ *Enter maximum transactions records to return* (default=1): ◆ *Enter name of file for response:*

Operation	Description
4: *Delete cached transactions*	Deletes the transactions stored in cache. ◆ *Enter position token* (default=0): ◆ *Enter event-id value of first transaction record* to delete (optional): ◆ *Enter number of transaction records to delete* (default=1):
99: *Exit*	Exits the current menu and takes you back to the Driver options.

Figure A-4 *Log Event Operations*

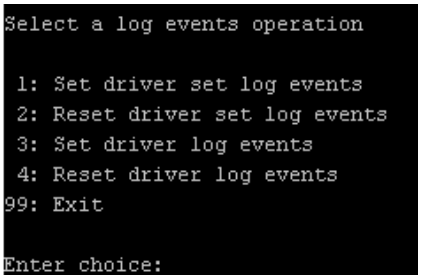


Table A-5 *Log Events Operations*

Operation	Description
1: *Set driver set log events*	Allows you to log driver set events through Novell Audit. There are 49 items you can select to log. See Table A-6 on page 96 for a list of these options. Type the number of the item you want to log. After the items are selected, enter 99 to accept the selections.
2: *Reset driver set log events*	Resets all of the log event options.
3: *Set driver log events*	Allows you to log driver events through Novell Audit. There are 49 items to select to log. See Table A-6 on page 96 for a list of these options. Type the number of the item you want to log. After the items are selected, enter 99 to accept the selections.
4: *Reset driver log events*	Resets all of the log event options.
99: *Exit*	Exits the log events operations menu.

Table A-6 *Driver Set and Driver Log Events*

Options
1: Status success
2: Status retry
3: Status warning
4: Status error
5: Status fatal
6: Status other
7: Query elements
8: Add elements
9: Remove elements
10: Modify elements
11: Rename elements
12: Move elements
13: Add-association elements
14: Remove-association elements
15: Query-schema elements
16: Check-password elements
17: Check-object-password elements
18: Modify-password elements
19: Sync elements
20: Pre-transformed XDS document from shim
21: Post input transformation XDS document
22: Post output transformation XDS document
23: Post event transformation XDS document
24: Post placement transformation XDS document
25: Post create transformation XDS document
26: Post mapping transformation <inbound> XDS document
27: Post mapping transformation <outbound> XDS document
28: Post matching transformation XDS document
29: Post command transformation XDS document
30: Post-filtered XDS document <Publisher>
31: User agent XDS command document

Options
32: Driver resync request
33: Driver migrate from application
34: Driver start
35: Driver stop
36: Password sync
37: Password request
38: Engine error
39: Engine warning
40: Add attribute
41: Clear attribute
42: Add value
43: Remove value
44: Merge entire
45: Get named password
46: Reset Attributes
47: Add Value - Add Entry
48: Set SSO Credential
49: Clear SSO Credential
50: Set SSO Passphrase
51: User defined IDs
99: Accept checked items

Table A-7 *Enter Table Title Here*

Options	Description
1: *Get available job definitions*	Allows you to select an existing job. *Enter the job number:* *Do you want to filter the job definitions by containment?* Enter Yes or No *Enter name of the file for response:* Examples: NetWare: `sys:\files\user.log` Windows: `c:\files\user.log` Linux: `/files/user.log`

Options	Description
2: *Operations on specific job object*	Allows you to perform operations for a specific job.

A.2 Command Line Mode

The command line mode allows you to use script or batch files. Table A-8 on page 98 lists the different options that are available.

To use the command line options, decide which items you want to use and string them together.

Example: `dxcmd -user admin.headquarters -host 10.0.0.1 -password n0vell -start test.driverset.headquarters`

This example command starts the driver.

Table A-8 *Command Line Options*

Option	Description
Configuration	
-user *<user name>*	Specify the name of a user with administrative rights to the drivers you want to test.
-host *<name or IP address>*	Specify the IP address of the server where the driver is installed.
-password *<user password>*	Specify the password of the user specified above.
-port *<port number>*	Specify a port number, if the default port is not used.
-q *<quiet mode>*	Displays very little information when a command is executed.
-v *<verbose mode>*	Displays detailed information when a command is executed.
-s *<stdout>*	Writes the results of the `dxcmd` command to `stdout`.
-? *<show this message>*	Displays the help menu.
-help *<show this message>*	Displays the help menu.
Actions	
-start *<driver dn>*	Starts the driver.
-stop *<driver dn>*	Stops the driver.
-getstate *<driver dn>*	Shows the state of the driver as running or stopped.
-getstartoption *<driver dn>*	Shows the startup option of the driver.
-setstartoption *<driver dn>* *<disabled|manual|auto>* *<resync|noresync>*	Sets how the driver starts if the server is rebooted. Sets whether the objects are to be resynchronized when the driver restarts.

Option	Description
-getcachelimit *<driver dn>*	Lists the cache limit set for the driver.
-setcachelimit *<driver dn> <0 or positive integer>*	Sets the cache limit for the driver.
-migrateapp *<driver dn> <filename>*	Processes an XML document that contains a query command. Create the XML document that contains a query command by using the Novell `nds.dtd` (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_dtd/data/dtdndsoverview.html#dtdndsoverview).
-setshimpassword *<driver dn> <password>*	Sets the application password. This is the password of the user account you are using to authenticate into the connected system with.
-clearshimpassword *<driver dn> <password>*	Clears the application password.
-setremoteloaderpassword *<driver dn> <password>*	Sets the Remote Loader password. The Remote Loader password is used to control access to the Remote Loader instance.
<clearremoteloaderpassword *<driver dn>*	Clears the Remote Loader password.
-sendcommand *<driver dn> <input filename> <output filename>*	Processes an XDS command document. Specify the XDS command document as the input file. Examples: NetWare: `sys:\files\user.xml` Windows: `c:\files\user.xml` Linux: `/files/user.log` Specify the output filename to see the results. Examples: NetWare: `sys:\files\user.log` Windows: `c:\files\user.log` Linux: `/files/user.log`
-sendevent *<driver dn> <input filename>*	Submits a document to the driver's Subscriber channel, bypassing the driver cache. The document is processed ahead of anything that might be in the cache at the time of the submission. It also means that the submission fails if the driver is not running.

Option	Description
-queueevent *<driver dn> <input filename>*	Submits a document to the driver's Subscriber channel by queuing the document in the driver cache. The document gets processed after anything that might be in the cache at the time of the submission. The submission won't fail if the driver isn't running.
-setlogevents *<dn> <integer ...>*	Sets Novell Audit log events on the driver. The integer is the option of the item to log. See Table A-6 on page 96 for the list of the integers to enter.
-clearlogevents *<dn>*	Clears all Novell Audit log events that are set on the driver.
-setdriverset *<driver set dn>*	Associates a driver set with the server.
-cleardriverset	Clears the driver set association from the server.
-getversion	Shows the version of Identity Manager that is installed.
-initdriver object *<dn>*	Performs an internal initialization of data on a new Driver object. This is only for testing purposes.
-setnamedpassword *<driver dn> <name> <password> [description]*	Sets named passwords on the driver object. You specify the name, the password, and the description of the named password.
-clearnamedpassword *<driver dn> <name>*	Clears a specified named password.
-startjob *<job dn>*	Starts the specified job.
-abortjob *<job dn>*	Aborts the specified job.
-getjobrunningstate *<job dn>*	Returns the specified job's running state.
-getjobenabledstate *<job dn>*	Returns the specified job's enabled state.
-getjobnextruntime *<job dn>*	Returns the specified job's next run time.
-updatejob *<job dn>*	Updates the specified job.
-clearallnamedpaswords *<driver dn>*	Clears all named passwords set on a specific driver.

If a command line is executed successfully, it returns a zero. If the command line returns anything other than zero, it is an error. For example 0 means success, and -641 means invalid operation. -641 is an eDirectory error code. Table A-9 on page 101 contains other values for specific command line options.

Table A-9 *Command Line Option Values*

Command Line Option	Values
-getstate	0- stopped
	1- starting
	2- running
	3- shutting down
	11- get schema
	Anything else that is returned is an error.
-getstartoption	0- disabled
	1- manual
	2- auto
	Anything else that is returned is an error.
-getcachelimit	0- unlimited
	Anything else that is returned is an error.
-getjobrunningstate	0- stopped
	1- running
	Anything else that is returned is an error.
-getjobenabledstate	0- disabled
	1- enabled
	2- configuration error
	Anything else that is returned is an error.
-getjobnextruntime	Return is the next scheduled time for the job in eDirectory time format (number of seconds since 00:00:00 Jan 1, 1970UTC).