NetIQ® Identity Manager

Administrator's Guide for Drivers

May 2024

Legal Notice

Copyright 2009 - 2024 Open Text.

The only warranties for products and services of Open Text and its affiliates and licensors (“Open Text”) are as may be set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Open Text shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

For additional information, such as certification-related notices and trademarks see (https://www.microfocus.com/en-us/legal).

Contents

About this Book and the Library **9**

1 Introduction to Drivers **11**

Understanding Drivers 11
How a Driver Works 11
Installing the Driver Files 12
Creating the Driver Object in Designer 13
Importing the Current Driver Packages 13
Installing the Driver Packages 14
Configuring the Driver Object 17
Deploying the Driver 17
Starting the Driver 18
Activating Drivers 19
Securing Driver Communication through HTTPS 19
Configuring the Publisher Channel 19
Configuring the Subscriber Channel 21

2 Upgrading an Existing Driver **23**

Upgrading the Installed Packages 23
Applying the Driver Patch 24
Prerequisites 24
Applying the Patch as a Root User 24
Applying the Patch as a Non-Root User 25

3 Deciding Whether to Use the Remote Loader **27**

Understanding Shims 27
Determining When to Use the Remote Loader 27
Understanding the Java Remote Loader 28
Installing the Remote Loader 28
Requirements 28
Supported Drivers 28
Installing Remote Loader 30
Configuring the Remote Loader and Drivers 30
Creating a Secure Connection to the Identity Manager Engine 31
Understanding the Configuration Parameters for the Remote Loader 34
Configuring the Remote Loader for Driver Instances on UNIX or Linux 48
Configuring the Remote Loader for Driver Instances on Windows 49
Configuring the Java Remote Loader for Driver Instances 52
Configuring the .NET Remote Loader for Driver Instances 53
Configuring Identity Manager Drivers to Work with the Remote Loader 56
Configuring Mutual Authentication with the Identity Manager Engine 57
Starting and Stopping the Remote Loader 66

Verifying the Configuration 70

4 Viewing Version Information 73

5 Backing Up Drivers 75

Exporting the Driver in Designer 75
Running the Driver in Factory Mode 75

6 Monitoring Driver Health 77

Creating a Driver Health Job 78
Modifying the Driver Health Job's Settings. 78
Modifying the Conditions for a Driver Health Configuration 79
Modifying the Actions for a Driver Health Configuration 82
Creating a Custom State 84
Memory Requirements for Driver Health 84

7 Viewing Driver Statistics 87

Viewing Statistics for an Individual Driver. 87
Viewing Statistics for a Driver Set 87

8 Managing Associations between Drivers and Objects 89

Associations. 89
No-Reference Associations. 89
Migration Between Associations 90
Association Actions in the Main Menu 90
Association Actions in the Driver Menu. 91
Tools for Managing Associations 92
Inspecting Objects 92
Inspecting Drivers 93

9 Managing Driver Cache Files 95

Viewing a Transaction 95
Viewing the Out of Band Sync Cache 96
Relocating the Event Cache File 96

10 Securely Storing Driver Passwords with Named Passwords 97

Using Designer to Configure Named Passwords. 97
Using Identity Console to Configure Named Passwords 97
Using Named Passwords in Driver Policies 98
Using the Policy Builder. 98
Using XSLT. 98
Using the DirXML Command Line Utility to Configure Named Passwords. 99
Creating a Named Password in the DirXML Command Line Utility 99
Removing a Named Password in the DirXML Command Line Utility. 99

11 Configuring Java Environment Parameters **101**

Using Identity Console to Configure the Java Environment Parameters 101
Using Designer to Configure the Java Environment Parameters 102

12 Rights Needed by a Driver on Identity Vault Objects **105**

13 Using the DirXML Command Line Utility **107**

Interactive Mode 107
Command Line Mode 117

14 Synchronizing Objects **123**

What Is Synchronization? 123
When Is Synchronization Done? 123
How Does the Identity Manager Engine Decide Which Object to Synchronize? 124
How Does Synchronization Work? 125
Scenario One 125
Scenario Two 127
Scenario Three 128

15 Association Statistics **129**

16 Enabling Out of Band Sync **131**

Enabling Out of Band Sync Using Designer 131
Specifying the Out of Band Sync Status Interval 131

17 Migrating and Resynchronizing Data **133**

18 Configuring Stronger Ciphers for SSL Communication **135**

Prerequisites 135
Configuring the Settings for Suite B Mode 136
Engine 136
Engine and Remote Loader Communication 136
Engine and Fan-Out Agent Communication 137
Identity Manager Drivers 137
Enabling Stronger Ciphers for SSL Communication 137
Verifying the Suite B Settings 138

19 Monitoring Identity Manager **141**

Viewing the Monitoring Statistics 142
Monitoring Identity Manager 145
Monitoring Job Statistics 145
Monitoring JVM Statistics 146
Monitoring a Driver Set 148
Monitoring User Application Statistics 149
Monitoring Workflow Statistics 154

20 Improving Driver Performance Using Subscriber Service Channel 161

Prerequisites 161
Configuring the Subscriber Service Channel 161
Subscriber Service Channel Support Compatibility 162
Tracing the Service Channel 162

21 Viewing Identity Manager Processes 165

Setting Permission for Monitoring Trace Files 165
Working with is-sensitive Attribute 165

22 Editing Driver Configuration Files 167

Variables in a Driver Configuration File 167
General Notes 168
Import Driver Notes 170
Flexible Prompting in a Driver Configuration File 171
Viewing the Informal Identity Manager Driver Configuration DTD 173

23 When and How to Use Global Configuration Values 175

Using GCVs to Adapt the Driver Configuration File to Changing Environments 175
When Not to Use GCVs 175
When to Use Driver Set GCVs, Driver GCVs, or Global Configuration Objects in Packages 176
Naming Convention for GCVs 176

24 Extending Custom Entitlements 179

Understanding the Resource Model 179
Configuring Custom Entitlements 181
Deploying Custom Entitlement Package for Identity Applications 181
Modifying Custom Entitlement Extension 182

25 Synchronizing Permission Changes from the Connected Systems 185

Planning Overview 186
Prerequisites 186
Understanding the Resource Model 187
Understanding the Components for PCRS 188
Preparing Your Environment 191
Setting Up Administrative User Accounts 191
Setting Up Administrative Passwords 192
Configuring Common Settings GCVs 192
Configuring Custom Entitlements 193
Understanding the PCRS Process 194
Viewing Permission Collection and Reconciliation Service Configuration Objects 197
Troubleshooting Permission Collection and Reconciliation Service Issues 198

26 Troubleshooting 201

Identity Manager Treats SYN_TIME values as Signed Integers Instead of Unsigned Integers 201
Attributes Are Removed After Synthetic Add or Optimization Operations 201

Using NetIQ Sentinel to Log Identity Manager Events . . . 201
Troubleshooting Driver Processes . . . 202
Driver Shim Errors . . . 202
Identity Manager Driver Errors . . . 205
Java Customization Errors . . . 208
Multiple Sync Events Occur When an Object is Moved in the Master Replica Server . . . 209
Rule Engine Does Not Honor the Mode Specified in the Set or Add Destination Attribute Value . . . 210
Reassociating a Driver Set Object with a Server . . . 210

A Data Synchronization Flow 213

The Identity Vault . . . 213
The Shim . . . 217
Channels . . . 218
Events and Commands . . . 218
Schema Mapping Policy . . . 219
Event Transformation Rule . . . 219
Publisher . . . 219
Subscriber . . . 220
Filter . . . 220
The Sync Attribute . . . 221
The Notify Attribute . . . 221
Add Processor . . . 221
Publisher . . . 221
Subscriber . . . 222
Matching Rule . . . 222
Publisher . . . 223
Subscriber . . . 223
Create Rule . . . 223
Publisher . . . 223
Subscriber . . . 224
Placement Rule . . . 224
Publisher . . . 224
Subscriber . . . 224
Command Transformation Rule . . . 225
Publisher . . . 225
Subscriber . . . 225
Rules, Policies, and Style Sheets . . . 225
Input Transform Rule . . . 227
Output Transform Rule . . . 228
Associations . . . 228
Synthetic Adds . . . 229
Merge Processing . . . 231

B Driver Properties 237

Driver Configuration . . . 237
Driver Module . . . 237
Authentication . . . 238
Startup Option . . . 239
Driver Parameters . . . 239
ECMAScript . . . 240

Global Configuration 240
Named Passwords 240
Engine Control Values 240
Log Level 244
Security Equals 245
Filter 245
Edit Filter XML 245
Misc/Trace 245
Excluded Objects 246
Driver Health Configuration 246
Driver Manifest 246
Driver Cache Inspector 246
Driver Inspector 247

C Understanding Identity Manager Trace 249

Prerequisites 249
Configuring Trace 250
Configuring Trace Levels 250
Understanding the Trace Messages 255
Interpreting Trace During Identity Manager Operations 265
When a Rule Executes 265
During Driver Startup 267
During Query Processing 269
When Direct Commands are Processed 273
During add-association Events 273
Cases with No Trace Excerpts 275

D The Cache Flush Parameter 277

About this Book and the Library

The *Identity Manager Driver Administration Guide* provides information about administration tasks that are common to all Identity Manager drivers.

Intended Audience

This guide is for administrators, consultants, and network engineers who require a high-level introduction to Identity Manager business solutions, technologies, and tools.

Other Information in the Library

For more information about the library for Identity Manager, see the Identity Manager documentation website (https://www.netiq.com/documentation/identity-manager-49/).

1 Introduction to Drivers

As part of your Identity Manager deployment, NetIQ provides Identity Manager drivers to connect information between popular business applications, directories, and databases. The business policies you implement using drivers can help to reduce management costs, increase productivity and security, and provide event reporting and auditing.

This guide provides conceptual, procedural, and reference information that is applicable to all Identity Manager drivers. For information that is specific to individual drivers, see the appropriate guide on the Identity Manager Drivers Documentation Website (https://www.netiq.com/documentation/identity-manager-49-drivers/).

This chapter includes the following sections:

- "Understanding Drivers" on page 11
- "Installing the Driver Files" on page 12
- "Creating the Driver Object in Designer" on page 13
- "Activating Drivers" on page 19
- "Securing Driver Communication through HTTPS" on page 19

Understanding Drivers

The Identity Manager engine processes all data changes that occur in the Identity Vault or a connected application. For events that occur in the Identity Vault, the engine processes the changes and issues commands to the application via the driver. For events that occur in the application, the engine receives the changes from the driver, processes the changes, and issues commands to the Identity Vault.

Drivers connect the Identity Manager engine to the applications. A driver has two basic responsibilities: reporting data changes (events) in the application to the Identity Manager engine and carrying out data changes (commands) submitted by the Identity Manager engine to the application. Drivers must be installed on the same server as the connected application.

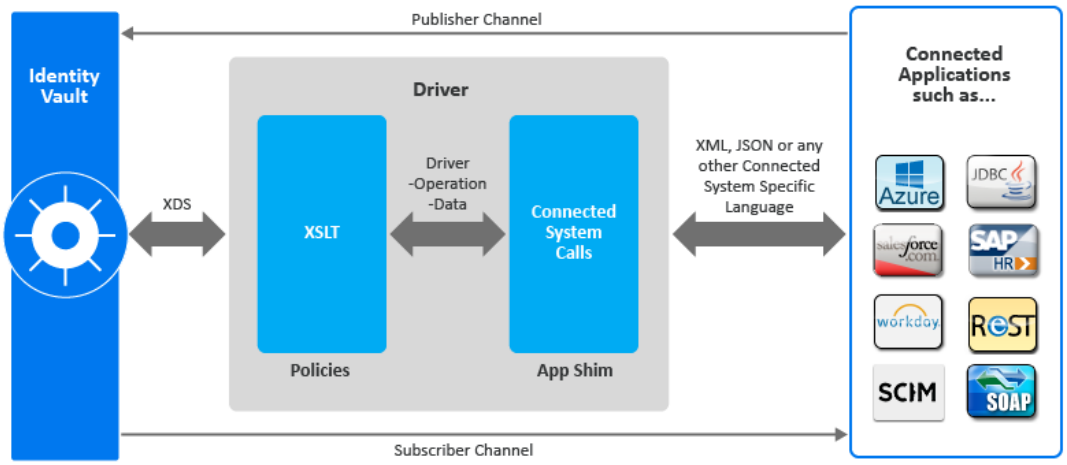
Identity Manager stores drivers and library objects in a container called a **driver set**. Only one driver set can be active on a server at a time. However, more than one server might be associated with one driver set. Also, a driver can be associated with more than one server at a time. However, the driver should be running on only one server at a time. The driver should be in a disabled state on the other servers. Any server that is associated with a driver set must have the Identity Vault installed on it.

For detailed information about Identity Manager components involved in data synchronization between the Identity Vault and the connected system, see Appendix A, "Data Synchronization Flow," on page 213.

How a Driver Works

Figure 1-1 illustrates the data flow between Identity Manager and a driver object.

Figure 1-1 Driver Dataflow



The Identity Manager engine uses XDS, a specialized form of XML, to represent events in the Identity Vault. Identity Manager passes the XDS to the driver policy, which consists of basic policies. The Driver uses a specialized form of XDS called `<driver-operation-data>`. The `<driver-operation-data>` element encapsulates the metadata and a payload.

When an event occurs in the Identity Vault, Identity Manager creates an XDS command to represent that event. Identity Manager passes the XDS command to the driver policy. The driver policy transforms that XDS command with an output transformation policy.

This output transformation policy generates the `<driver-operation-data>` that includes commands, URIs, methods, and payload information for the driver's request to the application shim. The shim then converts the `<driver-operation-data>` into a connected application's specific language for example, XML, JSON, etc., to communicate with the connected application.

When the request completes, the driver processes responses and reports the status of the completed operation to the Identity Manager engine or the Identity Vault.

On the Publisher channel, the driver shim receives the response from the connected application and converts the input into the `<driver-operation-data>` format. Then, using the input transformation policy, the driver converts the request to an XDS event and reports back to the Identity Vault.

Installing the Driver Files

To install the driver, you first need to install the driver files and then modify the driver configuration to suit your environment. Refer to the specific driver guide on the Identity Manager Drivers Documentation Website (https://www.netiq.com/documentation/identity-manager-49-drivers/) for the specific driver files to be installed.

Creating the Driver Object in Designer

The Designer tool helps you to create the driver object. You need to install the driver packages and then modify the configuration parameters. After you create and configure the driver, you need to deploy it to the Identity Vault and start it. Creating the driver object in Designer consists of the following tasks:

- "Importing the Current Driver Packages" on page 13
- "Installing the Driver Packages" on page 14
- "Configuring the Driver Object" on page 17
- "Deploying the Driver" on page 17
- "Starting the Driver" on page 18

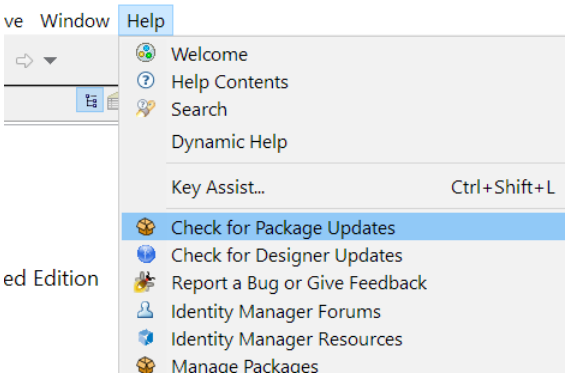
NOTE: NetIQ recommends that you use the new package management features provided in Designer to create the driver.

Importing the Current Driver Packages

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

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

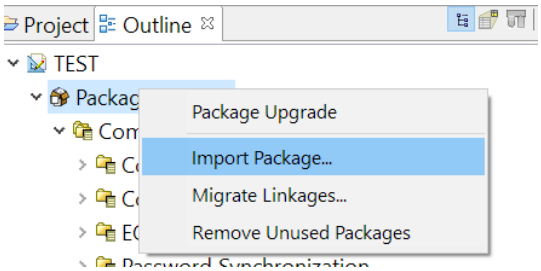
1. Open Designer.
2. In the toolbar, click **Help** > **Check for Package Updates**.



3. Click **OK** to update the packages

   or

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



6 Select the required driver package

or

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

By default, only the base packages are displayed. Deselect **Show Base Packages Only** to display all packages.

7 Click **OK** to import the selected packages, then click **OK** in the successfully imported packages message.

8 After the current packages are imported, continue with "Installing the Driver Packages" on page 14.

Installing the Driver Packages

After you have imported the current driver packages into the **Package Catalog**, you can install the driver packages to create a new driver, or update the existing driver package. Refer to the specific driver guides available in Identity Manager 4.9 Drivers Documentation (https://www.netiq.com/documentation/identity-manager-49-drivers/) page for the required packages to be installed.

To install driver packages, you must set-up Identity Vault and the driver set.

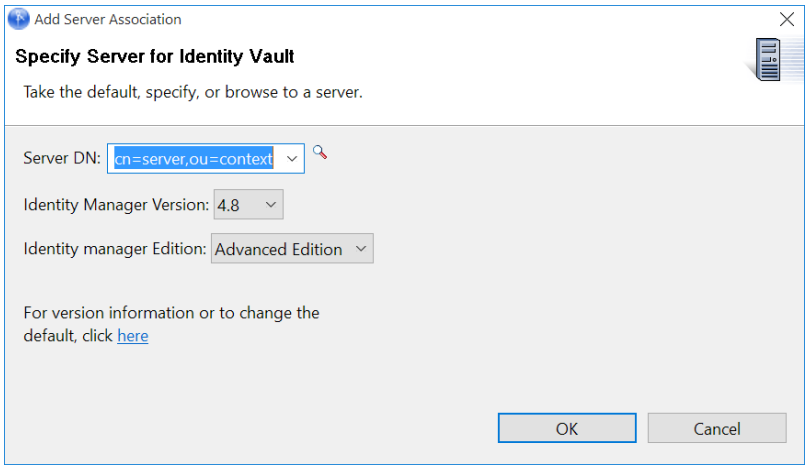
Setting Up the Identity Vault

If you are installing the driver for the first time you must setup the Identity Vault first for the driver to be configured. Perform the following steps to install an Identity Vault in the **Modeler** window.

1 In **Designer > Outline** view, open your project.

2 Right click project > **New** > **Identity Vault**, or drag and drop Identity Vault from the Palette to **Modeler** window.

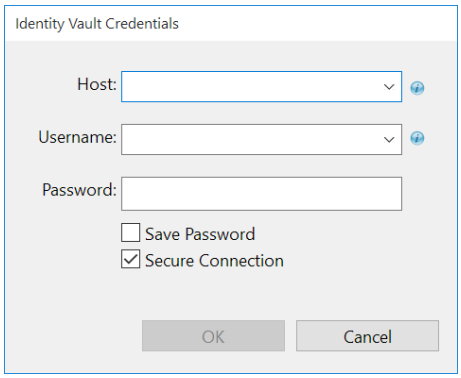
The **Add Server Association** screen appears.



3 In the **Add Server Association** screen, select the following field values and click **OK**.

- Server DN
- Identity Manager Version
- Identity Manager Edition

The **Identity Vault Credentials** window appears.



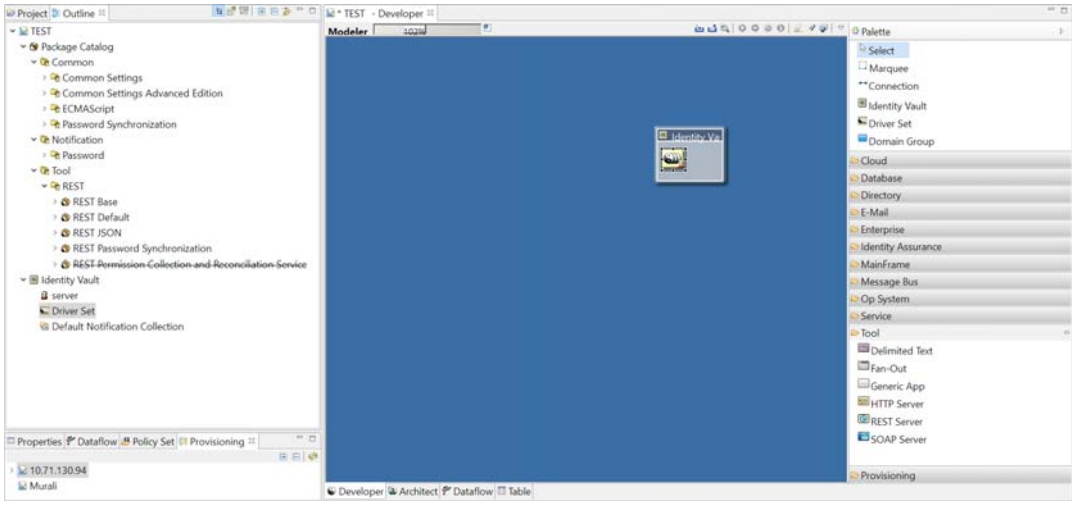
4 In Identity Vault Credentials window, enter:

Field	Description
Host	The identity vault hosting machine's IP address
Username	The name of the user, for example, Admin, if the user is an administrator.
Password	Password of user to log in to the identity vault

5 Select **Save Password**, if you want to save your password for easy logins in the future.

6 Click **OK**.

The Identity Vault and the Driver Set appears in the **Modeler** window as shown in the following image.



Setting Up a New Driver Object

After the Identity Vault is setup, you can add the required driver to connected to the Identity Vault. The drivers are categorized and available in the right pane under the respective categories.

1. In the right pane, drag and drop the driver object from the respective category tab to the **Modeler**.
2. In the **Driver Configuration Wizard**, select the base package.

   NOTE: You can only select one base package.

3. Click **Next**.
4. Select the optional features to install for the selected driver and click **Next**. The package dependencies window appears.
5. (Conditional) Click **OK** to install the package dependency listed.

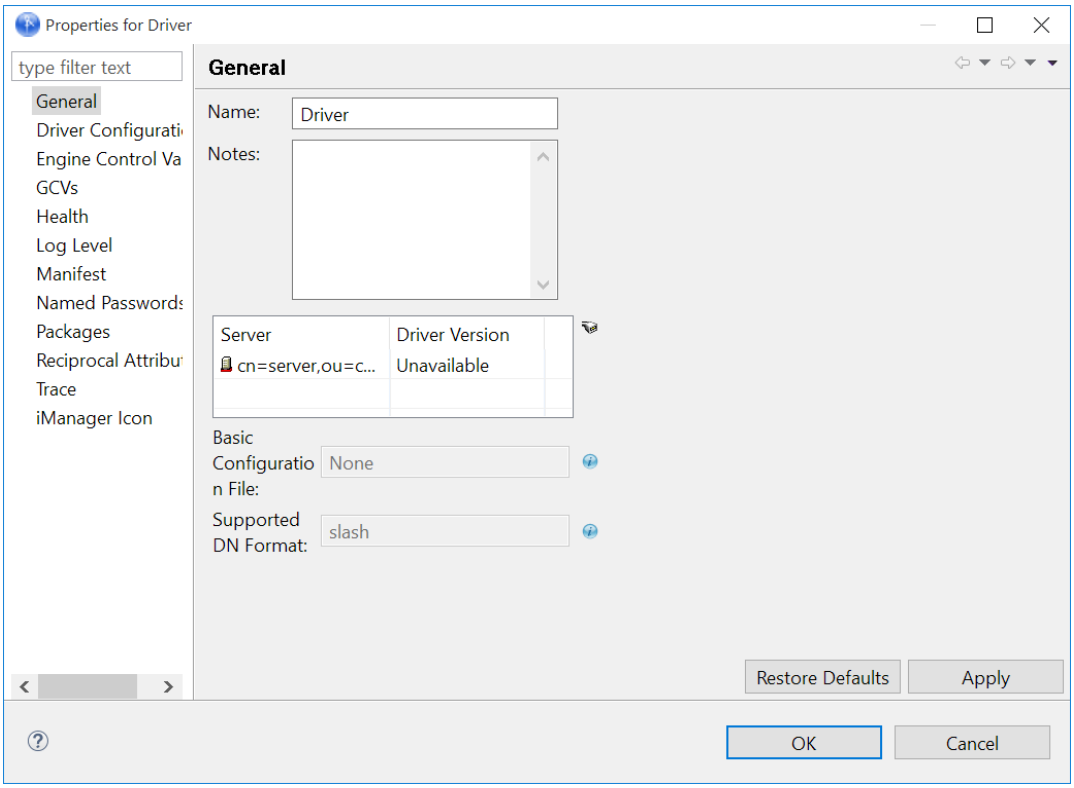
   NOTE: If there are any dependent packages associated with the selected package, you must install them to proceed.

6. On the **Driver Information** page, specify a name for the driver, then click **Next**. The driver configuration page appears.

   IMPORTANT: As the driver configuration parameters and their corresponding values are unique to each individual driver, you must refer to the driver specific configurations in the respective driver guides. For more information see, Identity Manager Drivers Documentation Website (https://www.netiq.com/documentation/identity-manager-49-drivers/) for driver specific guides.

Configuring the Driver Object

After the driver packages are installed, you need to configure the driver before it can run. You should complete the following tasks to configure the driver:



- **Configure the driver parameters:** There are many settings that can help you customize and optimize the driver. The settings are divided into categories such as Driver Configuration, Engine Control Values, and Global Configuration Values (GCVs). Although it is important for you to understand all of the settings, your first priority should be to review the Appendix B, "Driver Properties," on page 237. The Driver Parameters let you configure the publication method and other parameters associated with the Publisher channel.
- **Customize the driver policies and filter:** The driver policies and filter control data flow between the Identity Vault and the application. You should ensure that the policies and filters reflect your business needs.
- **Set Up a Secure HTTPS Connection:** The connection between the driver and the connected system can be configured to use a secure HTTPS connection rather than an HTTP connection.

After completing the configuration tasks you must deploy the driver.

Deploying the Driver

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

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

3 If you are authenticated to the Identity Vault, skip to Step 4; otherwise, specify the following information, then click **OK**.

Field	Description
Host	Specify the IP address or DNS name of the server hosting the Identity Vault.
Username	Specify the DN of the user object used to authenticate to the Identity Vault.
Password	Specify the user's password.

4 Read the deployment summary, then click **Deploy**.

5 Read the message, then click **OK**.

6 Click **Define Security Equivalence** to assign rights to the driver.

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

6a Click **Add**, then browse to and select the object with the correct rights.

6b Click **OK** twice.

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

7 Click **Exclude Administrative Roles** to exclude users that should not be synchronized.

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

7a Click **Add**, then browse to and select the user object you want to exclude, then click **OK**.

7b Repeat Step 7a for each object you want to exclude, then click **OK**.

8 Click **OK**.

9 Continue with the next section,"Starting the Driver" on page 18.

Starting the Driver

When a driver is created, it is stopped by default. To make the driver work, you must start the driver. Identity Manager is an event-driven system, so after the driver is started, it remains idle until an event occurs. You can use Identity Console or `dxevent` commands to start the driver.

To start the driver using Designer:

1 In Designer, open your project.

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

To start the driver using Identity Console:

1 Click the **IDM Administration** tab from the Identity Console home screen.

2 Click the specific driver set object on the right hand side of your computer screen to display all the drivers associated to it.

3 Click the Actions menu icon on the specific driver and select **Start Driver**.

IMPORTANT: When you start the driver for the first time, don't add new users to the Publisher channel until the first polling interval completes because the driver treats all users as existing users and stores them in the change cache without sending them to the Identity Manager engine. It sends the new users to the Identity Manager engine from the next polling interval. Therefore, ensure that new users are added to the Publisher channel after the first polling cycle completes.

Activating Drivers

Any new driver that is created requires a separate activation. After purchasing the integration module that the driver is associated, you will receive the activation details in your NetIQ Customer Center.

If you create a new driver in a driver set (that already includes an activated driver from its respective integration module), the new driver inherits activation from the driver set.

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

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

Securing Driver Communication through HTTPS

If the remote Web service that you are accessing allows HTTPS connections, you can configure the driver to take advantage of this increased security.

IMPORTANT: Only certificates from a Java keystore are accepted. Make sure that the keystore for the certificates is a Java keystore.

Configuring the Publisher Channel

The Publisher channel publishes the information from the connected application to the Identity Vault. To establish a secured connection for the Publisher Channel, you need a keystore or a KMO containing a certificate issued by the certificate authority that signed the server's certificate.

1 Create a server certificate in Identity Console:

   1a Click **Certificate Management** > **Server Certificate Management** options from the Identity Console landing page.

   1b Click **+ Create Server Certificate**.

   1c Specify a certificate nickname.

   1d Select **Standard** as the creation method, then click **Next**.

   1e Read through the parameters that will be used to create the server certificate. Click **OK**, then click **OK**.

2 Export a self-signed certificate from the certificate authority in eDirectory:

   2a In the **Roles and Tasks** view, click **Directory Administration > Modify Object**.

   2b Select your tree's certificate authority object, then click **OK**.

   It is usually found in the Security container and is named something like *TREENAME CA.Security*.

   2c Click **Certificate > Self Signed Certificate**.

   2d Click **Export**.

   2e When asked if you want to export the private key with the certificate, click **No**, then click **Next**.

   2f Based on the client to be accessing the Web service, select either **File in binary DER format** or **File in Base64 format** for the certificate, then click **Next**.

   If the client uses a Java-based keystore or trust store, then you can choose either format.

   2g Click **Save the exported certificate to a file**.

   2h Click **Save**, then browse to a known location on your computer.

   2i Click **Save**, then click **Close**.

3 Import the self-signed certificate into the client's trust store:

The steps to import the certificate vary depending on the client that connects to the Publisher channel's HTTPS listener. If the client uses a typical Java keystore, you can perform the following steps to create the keystore:

   3a Use the keytool executable that is included with any Java JDK.

   For more information about keytool, see Keytool - Key and Certificate Management Tool.

   3b Enter the following command at a command prompt:

```
keytool -import -file name_of_cert_file -trustcacerts -noprompt
-keystore filename -storepass password
```

   For example:

```
keytool -import -file tree_ca_root.b64 -trustcacerts -noprompt -
keystore dirxml.keystore -storepass novell
```

4 Configure the Publisher channel to use the server certificate you created in Step 1:

   4a In Identity Console, click the **IDM Administration** tab.

   4b Locate the driver set containing the driver, then click the driver object.

   4c Click **Configuration** > **Driver Parameters** > **Publisher Settings**.

   4d In the **KMO name** setting, specify the certificate nickname you used in Step 1.

5 Click **Apply**, then click **OK**.

Configuring the Subscriber Channel

The Subscriber channel sends information from the Identity Vault to the connected application. To establish a secured connection for the Subscriber channel, you need a trust store containing a certificate issued by the certificate authority that signed the server's certificate. See "Configuring the Publisher Channel" on page 19 for an example.

1. Make sure that you have a server certificate signed by a trusted certificate authority.
2. Import the certificate into your trust store or create a new trust store by entering the following command at the command prompt:

   ```
   keytool -import -file name_of_cert_file -trustcacerts -noprompt -
   keystore filename -storepass password
   ```

   For example:

   ```
   keytool -import -file tree_ca_root.b64 -trustcacerts -noprompt -
   keystore dirxml.keystore -storepass novell
   ```

   For more information about keytool, see Keytool - Key and Certificate Management Tool.
3. Configure the Subscriber channel to use the trust store you created in Step 2:
   - **3a** In Identity Console, click the **IDM Administration** tab.
   - **3b** Locate the driver set containing the driver, then click the driver object.
   - **3c** Click **Configuration** > **Driver Parameters** > **Subscriber Settings**.
   - **3d** In the **Keystore File** setting, specify the path to the trust store you created in Step 2.
4. Click **Apply**, then click **OK**.

2 Upgrading an Existing Driver

The driver upgrade process involves upgrading the installed driver packages and updating the driver files.

This section provides general instructions for updating a driver. For information about updating the driver to a specific version, search for that driver patch in the Software License and Download (https://sld.microfocus.com/mysoftware/index) portal and follow the instructions from the Readme file accompanying the driver patch release.

- "Upgrading the Installed Packages" on page 23
- "Applying the Driver Patch" on page 24

Upgrading the Installed Packages

Perform the following steps to upgrade the current installed package to the latest version.

1. Download the latest available packages.

   To configure Designer to automatically read the package updates when a new version of a package is available, click **Windows > Preferences > NetIQ > Package Manager > Online Updates** in Designer. For more information about managing packages, see Upgrading Installed Packages in the *NetIQ Designer for Identity Manager Administration Guide*.

2. Upgrade the installed packages.

   2a Open the project containing the driver.

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

   2c Click **Packages**.

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

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

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

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

   NOTE: Designer lists all versions available for upgrade.

   2g Click **Apply**.

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

   Depending on which package you selected to upgrade, you must fill in the required information to upgrade the package.

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

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

For detailed information, see the "Upgrading Installed Packages" in the *NetIQ Designer for Identity Manager Administration Guide*.

Applying the Driver Patch

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

- "Prerequisites" on page 24
- "Applying the Patch as a Root User" on page 24
- "Applying the Patch as a Non-Root User" on page 25

Prerequisites

Before installing the patch, complete the following steps:

1. Take a back-up of the current driver configuration.
2. (Conditional) If the driver is running with the Identity Manager engine, stop the Identity Vault and the driver instance.
3. (Conditional) If the driver is running with a Remote Loader instance, start the Remote Loader instance and the driver instance.
4. In a browser, navigate to the Software License and Download (https://sld.microfocus.com/mysoftware/index).
5. Under **Patches**, click **Search Patches**.
6. Specify the driver name in the search box.
7. Download and unzip the contents of the patch file to a temporary location on your server.

Applying the Patch as a Root User

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

1. Update the driver files:
   - **Linux:** To upgrade the existing RPM, log in as `root` and run the following command in a command prompt:

     ```
     rpm -Uvh <Driver Patch File Temporary Location>/linux/<driver filename>.rpm
     ```

     For example, `rpm -Uvh <IDM4.5_FanoutAgent_1110.zip>/linux/<driver filename>.rpm`
   - **Windows:** Navigate to the `<Extracted Driver Patch File Temporary Location>\windows` folder and copy the following files to `<IdentityManager installation>\NDS\lib` or `<IdentityManager installation>\RemoteLoader\lib` folder:
     - `<driver common filename>.jar`

- <driver shim filename>.jar
- <driver util filename>.jar

2 (Conditional) Start the Remote Loader instance.

3 (Conditional) Start the driver.

Applying the Patch as a Non-Root User

Perform the following steps if you are installing the patch as a non-root user.

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

If _db.000 is not present in this directory, the installation will not succeed.

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

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

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

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

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

3 Deciding Whether to Use the Remote Loader

The Remote Loader allows you to run Identity Manager drivers on connected systems that do not host the Identity Vault and Identity Manager engine. The .NET Remote Loader works on Windows-based systems only.

The Remote Loader is capable of hosting Identity Manager application shims contained in platform-specific files through JNI, as well as the more-common Identity Manager application shims contained in platform-agnostic JAR files. The Remote Loader can run on any platform. However, platform-specific shims must be run on their native platform (for example, `.so` files on Linux/Unix).

Understanding Shims

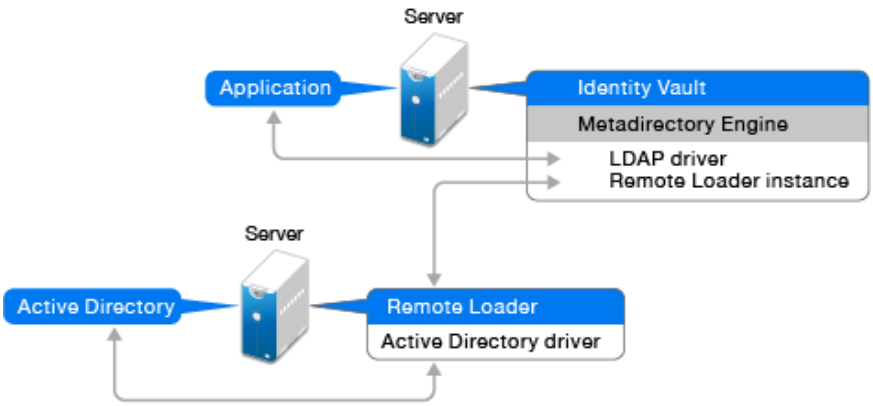
The Remote Loader uses shims to communicate with the application on a managed system. A shim is the file or files that contain the code to process the events that are synchronizing between the Identity Vault and the application. Before using the Remote Loader, you must configure the application shim to connect securely with the Identity Manager engine. You must also configure both the Remote Loader and the Identity Manager drivers. For more information, see “Configuring the Remote Loader and Drivers,”.

Determining When to Use the Remote Loader

You can install the Identity Manager engine, Identity Vault, and the driver shim on the same server. The Identity Manager engine runs as part of an eDirectory process. The Identity Manager drivers can run on the server with the Identity Manager. They also can run as part of the same process as the Identity Manager engine. However, in the following scenarios, you might want the Identity Manager driver to run as a separate process on the server that hosts the Identity Manager engine:

- To protect the Identity Vault from any exceptions encountered by the driver shim.
- To improve the performance of the server running the Identity Manager engine, by offloading driver commands to the remote application or database.
- To run additional drivers on servers that do not host the Identity Manager engine.

In these scenarios, the Remote Loader provides a communication channel between the Identity Manager engine and the driver. For example, you install an LDAP driver on the same server as the Identity Manager engine and the Identity Vault. Then you install the Active Directory (AD) driver on a different server with the Remote Loader. To allow the drivers to access the application and communicate with the Identity Vault, install the Remote Loader on both servers, as shown in the following figure.



NetIQ recommends that you use the Remote Loader configuration for use with your drivers where possible. Use the Remote Loader even in cases where the application is on the same server as the Identity Manager engine.

Understanding the Java Remote Loader

The Java Remote Loader provides the flexibility to load a driver shim on computers with UNIX or Linux servers that the native Remote Loader does not support. The Java Remote Loader is a Java application. You can use the Java Remote Loader with any publicly supported version of Java.

To open the application, run the shell script named `dirxml_jremote`. For more information, see "Configuring the Java Remote Loader for Driver Instances" on page 52.

Installing the Remote Loader

This section provides the following information:

Requirements

Each driver requires that the connected system be available and the relevant APIs are provided. Refer to the Identity Manager Driver documentation website (https://www.netiq.com/documentation/identity-manager-49-drivers/) for operating system and connected system requirements that are specific to each system.

Supported Drivers

NetIQ recommends that you use the Remote Loader configuration with your drivers where possible. Use the Remote Loader even in cases where the connected system is on the same server as the Identity Manager server engine.

When you run the driver shim in the Remote Loader configuration, the following advantages apply:

- Memory and processing isolation between driver shims allows for better performance and monitoring of the Identity Manager solution.
- Patching and upgrading the driver shim does not impact eDirectory or other drivers.

- Protects eDirectory from fatal issues that could occur in the driver shim.
- Distributes the load from the driver shims to other servers.

The following drivers support the Remote Loader capability:

- Active Directory
- Access Review
- ACF2
- Banner
- Blackboard
- Data Collection Services
- Delimited Text
- GoogleApps
- REST
- GroupWise (for 32-bit Remote Loader)
- JDBC
- JMS
- LDAP
- Linux/UNIX Settings
- Lotus Notes
- Managed System Gateway
- Manual Task Services
- Office 365
- Oracle EBS HRMS
- Oracle EBS TCA
- Oracle EBS User Management
- PeopleSoft 5.2
- Privileged User Management
- Remedy
- SalesForce.com
- SAP Business Logic
- SAP GRC (CMP only)
- SAP HR
- SAP Portal
- SAP User Management
- SCIM
- ServiceNow
- Integration Module V2.0 for Sentinel
- SharePoint

- SOAP
- Top Secret
- WorkOrder
- Workday

The following drivers do not support Remote Loader:

- Bidirectional eDirectory
- eDirectory
- Entitlements Services
- Null and Loopback
- Role Service
- User Application

Installing Remote Loader

Use the following information to install the Remote Loader for your platform.

Type of Remote Loader	See...
Installing Remote Loader on Linux	Installing Identity Manager in the *NetIQ Identity Manager Install and Upgrade Guide for Linux*.
Installing .NET Remote Loader on Windows	Installing Remote Loader in the *NetIQ Identity Manager Install and Upgrade Guide for Windows*.
Installing Java Remote Loader on Linux	Installing Java Remote Loader in the *NetIQ Identity Manager Install and Upgrade Guide for Linux*.
Installing Java Remote Loader on Windows	Installing Java Remote Loader in the *NetIQ Identity Manager Install and Upgrade Guide for Windows*.

Configuring the Remote Loader and Drivers

The Remote Loader can host the Identity Manager application shims contained in `.dll`, `.so`, or `.jar` files. The Java Remote Loader hosts only Java driver shims. It does not load or host a native (C++) driver shim.

Before using the Remote Loader, you must configure the application shim to connect securely with the Identity Manager engine. You must also configure both the Remote Loader and Identity Manager drivers. For more information about shims, see "Understanding Shims" on page 27.

- "Creating a Secure Connection to the Identity Manager Engine" on page 31
- "Understanding the Configuration Parameters for the Remote Loader" on page 34
- "Configuring the Remote Loader for Driver Instances on UNIX or Linux" on page 48
- "Configuring the Remote Loader for Driver Instances on Windows" on page 49
- "Configuring the Java Remote Loader for Driver Instances" on page 52
- "Configuring the .NET Remote Loader for Driver Instances" on page 53

- "Configuring Identity Manager Drivers to Work with the Remote Loader" on page 56
- "Configuring Mutual Authentication with the Identity Manager Engine" on page 57
- "Starting and Stopping the Remote Loader" on page 66
- "Verifying the Configuration" on page 70

Creating a Secure Connection to the Identity Manager Engine

You must ensure that data transfers securely between the Remote Loader and the Identity Manager engine. NetIQ recommends using Transport Layer Security/Secure Socket Layer (TLS/SSL) protocols for communication. To support TLS/SSL connections, you need an appropriate self-signed certificate in a keystore file. This section explains how to create, export, and store that certificate.

NOTE: Use the same version of SSL on the servers hosting the Identity Manager engine and the Remote Loader. If the versions of SSL on the server and the Remote Loader do not match, the server returns a `SSL3_GET_RECORD:wrong version number` error message. This message is only a warning, and communication between the server and Remote Loader is not interrupted. However, the error might cause confusion.

Understanding the Communication Process

The Remote Loader opens a server socket and listens for connections from the remote interface shim. The remote interface shim and the Remote Loader perform an SSL handshake to establish a secure channel. Then the remote interface shim authenticates to the Remote Loader. If the authentication of the remote interface shim succeeds, the Remote Loader authenticates to the remote interface shim. Only when both sides are satisfied that they are communicating with an authorized entity does synchronization traffic occur.

The process for establishing SSL connections between a driver and the Identity Manager engine depends on the type of driver:

- **For a native driver**, such as the Active Directory driver, point to a base64 encoded certificate. For more information, see "Managing Self-Signed Server Certificates" on page 32.
- **For a Java driver**, you must create a keystore. For more information, see "Creating a Keystore File when Using SSL Connections" on page 33.
- **For a .NET driver**, point to a base64 encoded certificate. For more information, see "Managing Self-Signed Server Certificates" on page 32.

  If the EC certificate is created with 521 as the key size, then you must add the NIST P-521 elliptic curve to the list of TLS ECC order on the Windows server where .NET Remote Loader is running. To do so, perform the following steps:

  1. Click **Run** > **gpedit.msc**.
  2. In the Local Group Policy Editor, navigate to **Computer Configuration** > **Administrative Templates** > **Network** > **SSL Configuration Settings**.
  3. Double-click **ECC Curve Order**.
  4. Set the value to **Enabled** and then add `NistP521` in the **ECC Curve Order** field.
  5. Save the changes and restart the Windows server.

NOTE: The Remote Loader allows for custom connection methods between the Remote Loader and the remote interface shim that is hosted on the Identity Manager server. To configure a custom connection module, see the documentation that comes with the module for information regarding what is expected and allowed in the connection string.

Managing Self-Signed Server Certificates

You can create and export a self-signed server certificate to ensure secure communication between the Remote Loader and the Identity Manager engine. For additional security, you can configure stronger ciphers for SSL communication as specified by Suite B. This communication requires the use of ECDSA (Elliptic Curve Digital Signature Algorithm) certificates for encrypting the data. When Suite B is enabled, Remote Loader uses TLS 1.2 as a communication protocol. For more information about Suite B, see Suite B Cryptography.

You can export a newly created certificate or use an existing certificate.

NOTE: When a server joins a tree, eDirectory creates the following default certificates:

- SSL CertificateIP
- SSL CertificateDNS
- Suite B compliant certificates

1. Log in to Identity Console.
2. To create a new certificate, complete the following steps:
   - **2a** Click on **Certificate Management > Server Certificate Management** options from the Identity Console landing page.
   - **2b** Click **+ Create Server Certificate**.
   - **2c** Select the server to own the certificate.
   - **2d** Specify a nickname for the certificate. For example, `remotecert`.

     NOTE: NetIQ recommends that you avoid using spaces in the certificate nickname. For example, use `remotecert` instead of `remote cert`.

     Also, make a note of the certificate nickname. This nickname is used for the KMO name in the driver's remote connection parameters.

   - **2e** Select the certificate creation method, then click **Next**.

     You have the following options:

     - **Standard:** This option creates a server certificate object using the largest possible key size and signs the public key certificate with your Organizational CA.
     - **Custom:** This option creates a server certificate object using the settings you specify. It allows you to set a number of customized settings for the Server Certificate object. Select this option to create ECDSA certificates for Suite B communication.
     - **Import:** This option creates a server certificate object using the keys and certificates from a PKCS12 (PFX) file. You can use this option in conjunction with the Export feature to backup and restore a Server Certificate or to move a Server Certificate object from one server to another.

2f Specify the certificate parameters.

2g Accept the rest of the certificate defaults.

2h Review the summary, click **Finish**, then click **Close**.

3 To export a certificate, complete the following steps:

3a Click on **Certificate Management > Server Certificate Management** options from the Identity Console landing page.

3b Select the appropriate server from the drop-down list.

3c Select the created certificate or the server created certificate from the list (for example, SSL CertificateDNS) and click the **Export Server Certificate** icon.

3d Select the **CA Certificate** as **OU=organization CA.O=TREEANAME** from the drop down menu.

3e Select the **Export Format** as **BASE64** from the drop down menu.

NOTE: When the Remote Loader is running on a Windows 2012 R2 64-bit server, the certificate must be in Base64 format. If you use the DER format, the Remote Loader fails to connect to the Identity Manager engine.

3f Click **OK**.

3g Click **Save the exported file**.

Creating a Keystore File when Using SSL Connections

To use SSL connections between a Java driver and the Identity Manager engine, you must create a keystore. A keystore is a Java file that contains encryption keys and, optionally, certificates. If you want to use SSL between the Remote Loader and the Identity Manager engine, and you are using a Java shim, you need to create a keystore file. The following sections explain how to create a keystore file:

- "Creating a Keystore on Any Platform" on page 33
- "Creating a Keystore on Linux" on page 34
- "Creating a Keystore on Windows" on page 34

Creating a Keystore on Any Platform

To create a keystore on any platform, you can enter the following at the command line:

```
keytool -import -alias trustedroot -file self-signed_certificate_name -
keystore filename -storepass keystorepass
```

The filename can be any name. For example, `rdev_keystore`.

Creating a Keystore on Linux

In Linux environments, use the `create_keystore` file, which is a shell script that calls the Keytool utility. The file is installed with rdxml, located by default in the *`install_directory`*`/dirxml/bin` directory. The create_keystore file is also included in the `dirxml_jremote.tar.gz` file, found in the `\dirxml\java_remoteloader` directory.

Perform the following actions to create the keystore:

1. Change to the directory in which you installed the JRE.

   ```
   % cd JAVA-HOME/bin
   ```

   For example: `% cd opt/netiq/common/jre/bin`

2. Set the path variable to include the location of the JRE at the command line.

   ```
   export PATH=opt/netiq/common/jre/bin
   ```

3. Create the keystore by entering the following at the command line:

   ```
   create_keystore self-signed_certificate_name keystorename
   ```

   For example, type one of the following

   ```
   create_keystore tree-root.b64 mystore
   create_keystore tree-root.der mystore
   ```

   The `create_keystore` script specifies a hard-coded password of “dirxml” for the keystore password. This is not a security risk because only a public certificate and public key are stored in the keystore.

Creating a Keystore on Windows

Run the Keytool utility, located by default in the `c:\novell\remoteloader\jre\bin` directory.

Understanding the Configuration Parameters for the Remote Loader

For the Remote Loader to work with a driver instance that hosts an Identity Manager application shim, you must configure the driver instance. For example, you must specify the connection and port settings for the instance. You can specify the settings from the command line, in a configuration file (UNIX or Linux), or in the Remote Loader Console (Windows). After the instance is running, you can use the command line to modify the configuration parameters or instruct the Remote Loader to perform a function. For example, you might want to open the trace window or unload the Remote Loader.

This section provides information about the configuration parameters. The explanation specifies whether a parameter can be sent from the command line to updated the Remote Loader while the instance is running.

For more information about configuring a new driver instance, see the following sections:

- **Linux and UNIX**: “Configuring the Remote Loader for Driver Instances on UNIX or Linux” on page 48
- **Windows**: “Configuring the Remote Loader for Driver Instances on Windows” on page 49.

Configuration Parameters for the Driver Instances in the Remote Loader

You can configure a driver instance from the command line or in a configuration file. NetIQ provides a sample file `config8000.txt` to help you configure the Remote Loader and drivers for use with your application shim. The sample file is located by default in the `/opt/novell/dirxml/doc` directory on Linux. On Windows, the sample file is located by default in the `C:\novell\remoteloader\<architecture(64bit/32bit>\` or `C:\Novell\remoteloader.NET` directory.

For example, the configuration file might include the following lines on Linux:

```
-commandport 8000
-connection "port=8090 rootfile=/dirxmlremote/root.pem"
-module $DXML_HOME/dirxmlremote/libcskeldrv.so.0.0.0
-trace 3
```

The configuration file might include the following lines on Windows:

```
-commandport 8000
-connection "port=8090"
-trace 4
-tracefile ./trace8000.log
-class com.novell.nds.dirxml.driver.delimitedtext.DelimitedTextDriver
```

Table 3-1 describes the options that enable you to configure the Remote Loader:

Table 3-1 *Remote Loader Options*

Option	Secondary Name	Parameter	Description
-assembly		path to driver DLL file	(Conditional) When using a .NET Remote Loader, specifies the path where the driver .dll is located. Ensure that the configuration file includes this parameter. For example: `-assembly C:\Novell\remoteloader.NET\DXMLMADDriver.dll`

Option	Secondary Name	Parameter	Description
-class	-cl	Java class name	(Conditional) When using a Java driver, specifies the Java class name of the Identity Manager application shim that you want to host. This options tells the application to use a Java keystore to read certificates. For example: `-class com.novell.nds.dirxml.driver.ldap.LDAPDriverShim -cl com.novell.nds.dirxml.driver.ldap.LDAPDriverShim` **NOTE:** ◆You cannot use this option if you specify a `-module` option. ◆ If you use the `tab` character as a delimiter in the `-class` option, the Remote Loader does not start automatically. Instead, you must manually start it. For the Remote Loader to start properly, you can use a space character instead of a `tab`. ◆ For more information about names that you can specify for this option, see "Understanding the Names for the Java -class Parameter" on page 47.
-commandport	-cp	port number	Specifies the TCP/IP port that the driver instance uses for control purposes. For example, `-commandport 8001` or `-cp 8001`. The default value is 8000. To use multiple driver instances with the Remote Loader on the same server, specify different connection ports and command ports for each instance. If the driver instance hosts an application shim, the command port is the port on which another instance communicates with the instance that is hosting the shim. If the driver instance sends a command to an instance that is hosting an application shim, the command port is the port on which the hosting instance is listening. When you send this parameter from the command line to an instance that hosts an application shim, the command port represents the port on which the hosting instance is listening. You can send this command when the Remote Loader is running.

Option	Secondary Name	Parameter	Description
-config	None	filename	Specifies a configuration file for the driver instance. For example: `-config config.txt` The configuration file can contain any command line options except `-config`. Options specified on the command line override options specified in the configuration file. You can send this command when the Remote Loader is running.

-description	-desc	short description	(Optional) Specifies a short description in string format, such as SAP, which the application uses for the title of the trace window and for audit logging. For example:
			`-description SAP`
			`-desc SAP`

-module	-m	modulename	(Conditional) When using a native drive, specifies the module containing the Identity Manager application shim that you want to host. This option tells the application to use a `rootfile` certificate. For example, for a native driver, type one of the following:
			`-module "c:\Novell\RemoteLoader\ADDriver.dll"`
			`-m "c:\Novell\RemoteLoader\ADDriver.dll"`
			NOTE: ◆You cannot use this option if you specify a `-class` option.
			◆ If you use the `tab` character as a delimiter in the `-module` option, the Remote Loader does not start automatically. Instead, you must manually start it. For the Remote Loader to start properly, you can use a space character instead of a `tab`.

Option	Secondary Name	Parameter	Description
-password	-p	password	Specifies the password for the driver instance when you issue commands that change settings or affect instance operation. You must specify the same password as the first password specified with setpasswords for the instance that you want to command. For example: `-password netiq4` If you do not send the password when issuing commands, the driver instance prompts you for the password. You can send this command when the Remote Loader is running.
-piddir	-pd	directory	Specifies the path to directory for the process id file (pidfile) used by the Remote Loader process. For example: `-piddir /var/opt/novell/dirxml/rdxml/data` The pidfile exists primarily for use by SysV-style init scripts. The default value is `/var/run`. Alternatively, the default value is the current directory, if the Remote Loader is run by a user without sufficient rights to open the pidfile for reading and writing in `/var/run`. This parameter is similar to `-datadir`.
-service	-serv	None, or install/uninstall	Specifies whether you want to configure an instance as a Win32 service. Valid values are `install` and `uninstall` plus the other parameters necessary to host an application shim. For example, you must include `-module` and might also include `-commandport` and the connection settings. This command simply installs or uninstalls the instance as a service. It does not start the service. You can send this command when the Remote Loader is running. However, you cannot use this command on rdxml or the Java Remote Loader.

Option	Secondary Name	Parameter	Description
-setpasswords	-sp	password password	Specifies the password for the driver instance and the password of the Identity Manager Driver object of the remote interface shim with which the Remote Loader communicates. You do not need specify a password. Instead, the Remote Loader prompts you for the passwords. However, if you specify the password for the Remote Loader, you must also specify the password for the Identity Manager Driver object associated with the remote interface shim on the Identity Manager engine server. To specify the passwords, use the following syntax: `-setpasswords` *`Remote_Loader_password driver_object_password`* For example: `-setpasswords netiq4 idmobject6` **NOTE:** Using this option configures the driver instance with the passwords specified but does not load a Identity Manager application shim or communicate with another instance.
-datadir	-dd	directory	Specifies the directory for data files that the Remote Loader uses. For example: `-datadir C:\novell\remoteloader` When you use this command, the Remote Loader changes its current directory to the specified directory. Trace files and other files that do not have an explicitly specified path will be created in this data directory.
-help	-h	None	Instructs the application to display the Help.

Option	Secondary Name	Parameter	Description

-connection	-conn	connection configuration string	Specifies the settings for connecting to the server hosting the Identity Manager engine that runs the Identity Manager remote interface shim. The default connection method is TCP/IP using SSL.
			
			To use multiple driver instances with the Remote Loader on the same server, specify different connection ports and command ports for each instance.
			
			Enter the connection settings in the following syntax:
			
			`-connection "`*`parameter parameter parameter`*`"`
			
			For example:
			
			`-connection "port=8091 fromaddress=198.51.100.0 rootfile=server1.pem keystore=ca.pem localaddress=198.51.100.0 hostname=198.51.100.0 kmo=remote driver cert"`

TCP/IP connection settings			Use the following parameters for specifying the settings for a TCP/IP connection:
address	None	IP_address	(Optional) Specifies whether the Remote Loader listens on a particular local IP address. This is useful if the server hosting the Remote Loader has multiple IP addresses and the Remote Loader must listen on only one of the addresses. The following values are valid: ◆ `address=address number` ◆ `address='localhost'` For example: `address=198.51.100.0` If you do not specify a value, the Remote Loader listens on all local IP addresses.

fromaddress	None	IP_address	Specifies the server from which the Remote Loader accepts connections. The application ignores connections from other addresses. Specify an IP address or the DNS name of the server. For example:
			
			`fromaddress=198.51.100.0`
			
			`fromaddress=testserver1.company.com`

Option	Secondary Name	Parameter	Description
handshaketimeout	None	Number of milliseconds	(Conditional) Applies when handshake timeouts occur with otherwise valid connections from the Identity Manager engine. Specifies the timeout period, in milliseconds, for the handshake between the Remote Loader and the Identity Manager engine. For example: `handshaketimeout=1000` You can specify an integer greater than or equal to zero. Zero means that the connection never times out. The default value is 1000 milliseconds.
hostname	None	IP_address or name of the server	Specifies the IP address or name of the server on which the Remote Loader runs. For example: `hostname=198.51.100.0`
secureprotocol	None	TLS version	Specifies the version of the TLS protocol that the Remote Loader uses to connect to the Identity Manager engine. For example: `secureprotocol=TLSv1_2` Identity Manager supports TLSv1 and TLSv1_2. By default, the Remote Loader uses TLSv1_2. To use TLSv1, specify this version in the parameter.
enforceSuiteB	None	True/False	(Conditional) Applies only when you want the Remote Loader to communicate with the Identity Manager engine using Suite B cryptographic algorithms. To use Suite B for communication, specify `true`. This communication is supported only on TLS 1.2 protocol. If you try to connect a Suite B-enabled engine with a Remote Loader that does not support TLSv1.2, the handshake fails and the communication is not established. For example, Remote Loader 4.5.3, which does not support TLS v1.2.
useMutualAuth	None	True/False	(Conditional) Applies only when want the Remote Loader and the Identity Manager engine to authenticate each other by verifying the public key certificate or digital certificate issued by the trusted Certificate Authorities (CAs) or self-signed certificates. For example: `useMutualAuth=true`

Option	Secondary Name	Parameter	Description
keystore		keystore filename	Specifies the file name of the Java keystore that contains the trusted root certificate of the issuer of the certificate that the remote interface shim uses. For example: `keystore=keystore filename` Usually, you specify the Certificate Authority of the tree that is hosting the remote interface shim.
kmo		kmo name	Specifies the key name of the Key Material Object containing the keys and certificate used for SSL connections. For example: `kmo=remote driver cert`
localaddress		IP_address	Specifies the IP address to which you want to bind the socket for client connection. For example: `localaddress=198.51.100.0`
port		decimal port number	Specifies the TCP/IP port on which the Remote Loader listens for connections from the remote interface shim. To specify the default port, enter `port=8090`.
rootfile		trusted root certificate filename	Specifies the name of the file that contains the trusted root certificate of the issuer of the certificate that the remote interface shim uses. The certificate file must be in Base 64 format (PEM). For example: `rootfile=trustedcert` Usually, the file will be the Certificate Authority of the tree that is hosting the remote interface shim.
storepass		storepass	Specifies password for the Java keystore that you entered for the `keystore` parameter. For example: `storepass=mypassword` For the Remote Loader to communicate with a Java driver, specify a key-value pair, using the following syntax: `keystore=`*`keystorename`* `storepass=`*`password`*
Java settings			Use the following parameters for specifying the Java settings:

Option	Secondary Name	Parameter	Description
-java	-j	None	(Conditional) Specifies that you want to set passwords for a Java driver shim instance. **NOTE:** If `-class` value is specified with `-setpasswords`, this option is not necessary.
-javadebugport	-jdp	Port number	Instructs the instance to enable Java debugging on the specified port. For example: `-javadebugport 8080` Use this command when developing Identity Manager application shims. You can send this command when the Remote Loader is running.

-javaparam	-jp		Specifies the parameters for the Java environment. Enter the Java environment parameters in the following syntax:
			
			`-javaparam` *`parameter`*
			`-jp` *`parameter`*
			`-jp` *`parameter`*
			
			NOTE: Do not use this parameter with the Java Remote Loader.
			
			To specify multiple values for an individual parameter, enclose the parameter in quotation marks. For example:
			
			`-javaparam DHOST_JVM_MAX_HEAP=512M`
			`-jp DHOST_JVM_MAX_HEAP=512M`
			`-jp "DHOST_JVM_OPTIONS=-Dfile.encoding=utf-8 -Duser.language=en"`

Java environment settings			Use the following parameters for setting the Java environment:
DHOST_JVM_ADD_CLASSPATH			Specifies additional paths for the JVM to search for package (`.jar`) and class (`.class`) files.

Option	Secondary Name	Parameter	Description
DHOST_JVM_INITIAL_HEAP			Specifies the initial (minimum) JVM heap size in decimal number of bytes. Use a numeric value followed by G, M, or K representing the byte type. For example: `100M` If you do not specify a byte type, the size defaults to bytes. Using this parameter is the same as using the java `-Xms` command. This parameter has precedence over the driver set attribute option. Increasing the initial heap size can improve startup time and throughput performance.
DHOST_JVM_MAX_HEAP			Specifies the maximum JVM heap size in decimal number of bytes. Use a numeric value followed by G, M, or K representing the byte type. For example: `100M` If you do not specify a byte type, the size defaults to bytes. This parameter has precedence over the driver set attribute option.
DHOST_JVM_OPTIONS			Specifies the arguments that you want to use when starting the JVM instance of the driver. Use a space to separate each option string. For example: `-Xnoagent -Xdebug -Xrunjdwp: transport=dt_socket,server=y, address=8000` The driver set attribute option has precedence over this parameter. This environment variable is tacked on to the end of driver set attribute option. For more information about valid options, see the JVM documentation.
Trace file settings			(Conditional) When hosting an Identity Manager application shim, specifies the settings for a trace file that contains informational messages from both the Remote Loader and the driver for this instance. Add the following parameters to the configuration file:

Option	Secondary Name	Parameter	Description
-trace	-t	integer	Specifies the level of messages that you want displayed in a trace window. For example: `-trace 3` Trace levels for the Remote Loader correspond to those used on the server hosting the Identity Manager engine.
-tracefile	-tf	filename	Specifies the path to a file where trace messages are logged. You must specify a unique trace file for each driver instance running on a particular computer. For example: `-tracefile /home/trace.txt` on Linux `-tracefile c:\temp\trace.txt` on Windows The application writes messages to the file if the `-trace` parameter is greater than zero. The trace window does not need to be open for messages to be written to the file.
-tracefilemax	-tfm	size	Specifies a limit to the size of the trace file for this instance. Specify the value in kilobytes, megabytes, or gigabytes, using the abbreviation for the byte type. For example: ◆ `-tracefilemax 1000K` ◆ `-tf 100M` ◆ `-tf 10G` **NOTE:** ◆When you add this option to the configuration file, the application uses the specified name for the tracefile and includes up to 9 “roll-over” files. Each file size is 1/10th of the total size specified. The roll-over files are named using the base of the main trace filename plus _n, where n is 1 through 9. ◆ If the trace file data is larger than the specified maximum when the Remote Loader is started, the trace file data remains larger than the specified maximum until roll-over is completed through all 10 files.

Option	Secondary Name	Parameter	Description
-tracechange	-tc	integer	(Conditional) When you have an existing driver instance that hosts an application shim, specifies a new level of informational messages. Trace levels correspond to those used on the Identity Manager server. For example: `-trace 3` You can send this command when the Remote Loader is running.
-tracefilechange	-tfc	None, or filename	(Conditional) When you have an existing driver instance that hosts an application shim, instructs that instance to use a trace file or to close a file already in use and change to this new file. For example: `-tracefilechange \temp\newtrace.txt` You can send this command when the Remote Loader is running.
Certificate password settings			(Conditional) Only when `useMutualAuth` is set to true in the configuration file. Use the following parameters for specifying the Certificate password settings:
-keystorepassword	-ksp	password	Specifies the keystore password to enable mutual authentication for Java Remote Loader drivers only.
-keypassword	-kp	password	Specifies the key password to enable mutual authentication for both Java and native Remote Loader drivers.
-unload	-u	None	Instructs the driver instance to unload. If the Remote Loader is running as a Win32 Service, this command stops the service. You can send this command when the Remote Loader is running.
-window	-w	On/Off	Instructs the application to turn on or off the trace window for a driver instance. Valid values are `on` and `off`. For example: `-window on` You can send this command when the Remote Loader is running. You cannot use this command with the Java Remote Loader.

Option	Secondary Name	Parameter	Description
-wizard	-wiz	None	Launches the Configuration Wizard for the Remote Loader. You can also launch the wizard by running `dirxml_remote.exe` with no command line parameters. If you run this command and also specify a configuration file (`-config` option), the wizard starts with the values from the configuration file. You can use the wizard to change the configuration without editing the configuration file directly. For example: `-wizard -config config.txt` You cannot use this command with the Java Remote Loader.

Understanding the Names for the Java -class Parameter

When you use the `-class` parameter to configure a driver instance for the Remote Loader and Java Remote Loader, you must specify the Java class name of the Identity Manager application shim that you want to host.

Java Class Name	Driver
com.novell.nds.dirxml.driver.dcsshim.DCSShim	Driver for Data Collection Service
com.novell.nds.dirxml.driver.delimitedtext.DelimitedTextDriver	Delimited Text Driver
be.opns.dirxml.driver.ars.arsremedydrivershim.ARSDriverShim	Driver for Remedy ARS
com.novell.nds.dirxml.driver.entitlement.EntitlementServiceDriver	Entitlements Service Driver
com.novell.gw.dirxml.driver.rest.shim.GWdriverShim	GroupWise 2014 Driver
com.novell.idm.drivers.idprovider.IDProviderShim	ID Provider Driver
com.novell.nds.dirxml.driver.jdbc.JDBCDriverShim	JDBC Driver
com.novell.nds.dirxml.driver.jms.JMSDriverShim	JMS Driver
com.novell.nds.dirxml.driver.ldap.LDAPDriverShim	LDAP Driver
com.novell.nds.dirxml.driver.loopback.LoopbackDriverShim	Loopback Driver
com.novell.nds.dirxml.driver.ebs.user.EBSUserDriver	Oracle User Management Driver
com.novell.nds.dirxml.driver.ebs.hr.EBSHRDriver	Oracle HR Driver
com.novell.nds.dirxml.driver.ebs.tca.EBSTCADriver	Oracle TCA Driver
com.novell.nds.dirxml.driver.msgateway.MSGatewayDriverShim	Managed System Gateway Driver
com.novell.nds.dirxml.driver.manualtask.driver.ManualTaskDriver	Manual Task Driver
com.novell.nds.dirxml.driver.nisdriver.NISDriverShim	NIS Driver

Java Class Name	Driver
com.novell.nds.dirxml.driver.notes.NotesDriverShim	Notes Driver
com.novell.nds.dirxml.driver.psoftshim.PSOFTDriverShim	PeopleSoft Driver
com.netiq.nds.dirxml.driver.pum.PUMDriverShim	Privileged User Management Driver
com.novell.nds.dirxml.driver.salesforce.SFDriverShim	Salesforce Driver
com.novell.nds.dirxml.driver.SAPHRShim.SAPDriverShim	SAP HR Driver
com.novell.nds.dirxml.driver.sap.portal.SAPPortalShim	SAP Portal Driver
com.novell.nds.dirxml.driver.sapumshim.SAPDriverShim	SAP User Management Driver
com.novell.nds.dirxml.driver.soap.SOAPDriver	SOAP Driver
com.novell.idm.driver.ComposerDriverShim	User Application
com.novell.nds.dirxml.driver.workorder.WorkOrderDriverShim	WorkOrder Driver

Configuring the Remote Loader for Driver Instances on UNIX or Linux

The Remote Loader can host the Identity Manager application shims contained in `.dll`, `.so`, or `.jar` files. For the Remote Loader to run on a UNIX or Linux computer, the application needs a configuration file such as `LDAPShim.txt` for each driver instance. You can also create or edit a configuration file by using command line options.

By default, the Remote Loader connects to the Identity Manager engine through TCP/IP using TLS/SSL protocols. The default TCP/IP port for this connection is 8090. You can run multiple driver instances with the Remote Loader on the same server. Each instance hosts a separate Identity Manager application shim instance. To use multiple instances of the Remote Loader on the same server, specify different connection ports and command ports for each instance.

NOTE: ◆The configuration file can contain any command line options except `-config`.

- When adding parameters to the configuration file, you can use the long form or a short form of the parameter. For example, `-description` or `-desc`.
- The following procedure lists the long form first, followed by the short form in parentheses. For example `-description` *`value`* (`-desc` *`value`*).
- For more information about the parameters used in this section, see "Understanding the Configuration Parameters for the Remote Loader" on page 34.

To create a configuration file:

1. In a text editor, create a new file.

   NetIQ provides a sample file `config8000.txt` to help you configure the Remote Loader and drivers for use with your application shim. The sample file is located by default in the `/opt/novell/dirxml/doc` directory.

2. Add the following configuration parameters to the file:
   - -description (optional)

- -commandport
- connection parameters:
  - port (mandatory)
  - address
  - fromaddress
  - handshaketimeout
  - rootfile
  - keystore
  - localaddress
  - hostname
  - kmo
  - secureprotocol
  - enforceSuiteB
  - useMutualAuth
- trace file parameters (optional):
  - -trace
  - -tracefile
  - -tracefilemax
- -javaparam
- -class or -module

For more information about specifying values for these parameters, see"Understanding the Configuration Parameters for the Remote Loader" on page 34.

3 Save the file.

For the Remote Loader to start automatically when your computer starts, save the file to the `/etc/opt/novell/dirxml/rdxml` directory.

Configuring the Remote Loader for Driver Instances on Windows

The Remote Loader can host the Identity Manager application shims contained in `.dll`, `.so`, or `.jar` files. For the Remote Loader to run, the application needs a configuration file, such as `LDAPShim.txt`. The Remote Loader Console utility (the Console) helps you manage all instances of Identity Manager drivers running on the Windows server. You can start, stop, add, remove, and edit each instance of a Remote Loader. The installation program for the Remote Loader also installs the Console.

If you are upgrading, the Console detects and imports existing driver instances. For a driver to be automatically imported, its configuration file must be stored in the Remote Loader directory, located by default at `c:\novell\remoteloader`. You can then use the Console to manage the remote drivers.

You can use the command line or the Remote Loader Console to configure the Remote Loader to recognize a driver on Windows. For more information about using the command line, see "Understanding the Configuration Parameters for the Remote Loader" on page 34.

This section provides instructions for the following activities:

- "Creating a New Driver Instance in the Remote Loader on Windows" on page 50
- "Modifying an Existing Driver Instance in the Remote Loader on Windows" on page 52

Creating a New Driver Instance in the Remote Loader on Windows

1 Open the Remote Loader Console.

NOTE: During installation, if you selected to create a shortcut for the Console, use the `Identity Manager Remote Loader Console` icon on the desktop. Otherwise, run `rlconsole.exe` located by default in `C:\novell\remoteloader\`*`nn`*`bit`.

---

2 To add an instance of your driver on this server, click **Add**.

3 For **Description**, provide a short name to represent the instance.

The Console uses this information in the default value for **Config File**.

4 For **Driver**, select the Java class name.

NOTE: To use the Active Directory driver, select **ADDriver.dll**. For more information about the class names for each driver, see "Understanding the Names for the Java -class Parameter" on page 47.

---

5 For **Config File**, specify the path to the file where Remote Loader stores its configuration parameters. The default value is `C:\novell\remoteloader\`*`nn`*`bit\`*`Description`*`-config.txt`.

6 Specify passwords for the Remote Loader and driver object.

7 (Optional) To use a TLS/SSL connection between the Remote Loader and the Identity Manager engine server, complete the following steps:

7a Select **Use an SSL Connection**.

NOTE: NetIQ recommends using the same version of SSL on both the Identity Manager engine server and the Remote Loader. If the versions of SSL on the server and the Remote Loader do not match, the server returns a "`SSL3_GET_RECORD:wrong version number`" error message. This message is only a warning, and communication between the server and Remote Loader is not interrupted. However, the error might cause confusion.

7b For **Trusted Root File** (base64 format file), specify the exported self-signed certificate from the eDirectory tree's Organization Certificate Authority. For more information, see "Creating a Secure Connection to the Identity Manager Engine" on page 31 and "Understanding the Configuration Parameters for the Remote Loader" on page 34.

8 (Optional) To configure the trace file for the Remote Loader, complete the following steps:

NOTE: NetIQ recommends using the trace functionality only for troubleshooting issues. Having the trace enabled reduces the performance of the Remote Loader. Do not leave the trace enabled in production.

8a For **Trace Level**, specify a value greater than zero that defines the level of informational messages from both the Remote Loader and the driver that you want display in a trace window. Values 1 to 4 are pre-defined by the Console. To create your own message types, specify a value of 5 or higher.

The most common setting is trace level 3, which provides general processing, XML documents, and Remote Loader messages.

8b For **Trace File**, specify the path to a file where trace messages are logged. For example, `C:\novell\remoteloader\64bit\Test-Delimited-Trace.log`.

You must specify a unique trace file for each driver instance running on a particular computer. Trace messages are written to the trace file only if the trace level is greater than zero.

8c For **Maximum Disk Space Allowed for all Trace Logs (Mb)**, specify an approximate value for the most disk space that the trace file for this instance can occupy.

9 (Optional) To allow the Remote Loader to start automatically when the computer starts, select **Establish Remote Loader Service for this driver instance**.

NOTE: If the SSL connection fails due to `handshaketimeout` when Remote Loader establishes connection with Identity Manager engine then, update the default `handshaketimeout` variable to 10000 and restart both driver and remote loader.

10 (Conditional) To modify the parameters for Java configuration, complete the following steps:

10a Select **Advanced**.

10b For **Classpath**, specify the paths for the JVM to search for package (`.jar`) and class (`.class`) files.

This parameter functions the same as the `java -classpath` command.

10c For **JVM Options**, specify the options that you want to use when starting the JVM instance of the driver.

10d Specify the initial and maximum heap size for the JVM instance in MB.

10e For Suite B communication, specify `enforceSuiteB=true`. This communication is supported only on TLS 1.2 protocol.

For more information, see "Creating a Secure Connection to the Identity Manager Engine" on page 31 and "Understanding the Configuration Parameters for the Remote Loader" on page 34.

10f Click **OK**.

11 (Optional) To allow the Remote Loader to use the secure protocol while connecting to the Identity Manager engine, specify the secure protocol version in the Remote Loader configuration file. For example: `secureprotocol=TLSv1_2`

For more information, see "Understanding the Configuration Parameters for the Remote Loader" on page 34.

NOTE: Skip this step if you already configured the secure protocol version on the driver.

12 (Optional) To allow the Remote Loader communication to use the protocols specified by Suite B, specify `enforceSuiteB=true` in the Remote Loader configuration file. This communication is supported only on TLS 1.2 protocol.

For more information, see "Understanding the Configuration Parameters for the Remote Loader" on page 34.

NOTE: Skip this step if you already enabled Suite B communication on the driver.

---

13 Click **OK**.

Modifying an Existing Driver Instance in the Remote Loader on Windows

1 In the Remote Loader Console, select the driver instance from the **Description** column.

2 Click **Stop**.

3 Enter the password for the Remote Loader, then click **OK**.

4 Click **Edit**.

5 Modify the configuration information. For more information about each parameter, see "Creating a New Driver Instance in the Remote Loader on Windows" on page 50.

6 To save the changes, click **OK**.

Configuring the Java Remote Loader for Driver Instances

The Java Remote Loader hosts only Java driver shims. It does not load or host a native (C++) driver shim.

To configure a new instance for the Java Remote Loader on Linux platforms, complete the following steps. For more information about the parameters used in this section, see "Understanding the Configuration Parameters for the Remote Loader" on page 34.

1 In a text editor, create a new file.

NetIQ provides a sample file `config8000.txt` to help you configure the Remote Loader and drivers for use with your application shim. The sample file is located at `/opt/novell/dirxml/doc/`.

2 Add the following parameters to the new configuration file:

- -description (optional)
- -class or -module

  For example, `-class com.novell.nds.dirxml.driver.ldap.LDAPDriverShim`
- -commandport
- connection parameters:
  - port (mandatory)
  - address
  - fromaddress
  - handshaketimeout
  - rootfile

    - keystore
    - localaddress
    - hostname
    - kmo
    - secureprotocol
    - enforceSuiteB
    - useMutualAuth
  - -java (conditional)
  - -javadebugport
  - -password
  - -service
  - -setpasswords
  - trace file parameters (optional):
    - -trace
    - -tracefile
    - -tracefilemax

NOTE: For more information about the parameters, see "Understanding the Configuration Parameters for the Remote Loader" on page 34.

3 Save the new configuration file.

  For the Remote Loader to start automatically when your computer starts, save the file to the `\jremote` directory.

4 Open a command prompt.

5 At the prompt, enter `-config` *`filename`*, where *filename* is the name of the new configuration file. For example:

```
dirxml_jremote -config <configFile> -service
```

  This starts the Java Remote Loader service and opens a trace window.

6 (Optional) To stop the driver service, go to Services, then stop the service.

Configuring the .NET Remote Loader for Driver Instances

The Remote Loader can host the Identity Manager application shim contained in `.dll` file. For the Remote Loader to run, the application needs a configuration file, such as `LDAPShim.txt`. The Remote Loader Console utility (the Console) helps you manage all instances of Identity Manager drivers running on the server. You can start, stop, add, remove, and edit each instance of a Remote Loader. The installation program for the Remote Loader also installs the Console.

If you are upgrading, the Console detects and imports existing driver instances. For a driver to be automatically imported, its configuration file must be stored in the Remote Loader directory, located by default at `c:\novell\remoteloader.net`. You can then use the Console to manage the remote drivers.

You can use the command line or the Remote Loader Console to configure the Remote Loader to recognize a driver. For more information about using the command line, see "Understanding the Configuration Parameters for the Remote Loader" on page 34.

This section provides instructions for the following activities:

- "Creating a New Driver Instance in the .NET Remote Loader" on page 54
- "Modifying an Existing Driver Instance in the .NET Remote Loader" on page 55

Creating a New Driver Instance in the .NET Remote Loader

1 Open the Remote Loader Console.

NOTE: During installation, if you selected to create a shortcut for the Console, use the `Identity Manager Remote Loader Console` icon on the desktop. Otherwise, run the `rlconsole.exe` located by default in `C:\novell\remoteloader.net`.

---

2 To add an instance of your driver on this server, click **Add**.

3 For **Description**, provide a short name to represent the instance.

The Console uses this information in the default value for **Config File**.

4 For **Driver**, select the appropriate driver.dll.

5 For **Config File**, specify the path to the file where Remote Loader stores its configuration parameters. The default value is `C:\novell\remoteloader.net\`*`Description`*`-config.txt`.

6 Specify passwords for the Remote Loader and driver object.

7 (Optional) To use a TLS/SSL connection between the Remote Loader and the Identity Manager engine server, complete the following steps:

7a Select **Use an SSL Connection**.

NOTE: NetIQ recommends using the same version of SSL on both the Identity Manager engine server and the Remote Loader. If the versions of SSL on the server and the Remote Loader do not match, the server returns a "`SSL3_GET_RECORD:wrong version number`" error message. This message is only a warning, and communication between the server and Remote Loader is not interrupted. However, the error might cause confusion.

7b For **Trusted Root File** (base64 format file), specify the exported self-signed certificate from the eDirectory tree's Organization Certificate Authority. For more information, see "Creating a Secure Connection to the Identity Manager Engine" on page 31 and "Understanding the Configuration Parameters for the Remote Loader" on page 34.

8 (Optional) To configure the trace file for the Remote Loader, complete the following steps:

NOTE: NetIQ recommends using the trace functionality only for troubleshooting issues. Having the trace enabled reduces the performance of the Remote Loader. Do not leave the trace enabled in production.

8a For **Trace Level**, specify a value greater than zero that defines the level of informational messages from both the Remote Loader and the driver that you want display in a trace window. Values 1 to 4 are pre-defined by the Console. To create your own message types, specify a value of 5 or higher.

The most common setting is trace level 3, which provides general processing, XML documents, and Remote Loader messages.

8b For **Trace File**, specify the path to a file where trace messages are logged. For example, `C:\novell\remoteloader.net\Test-Delimited-Trace.log`.

You must specify a unique trace file for each driver instance running on a particular computer. Trace messages are written to the trace file only if the trace level is greater than zero.

8c For **Maximum Disk Space Allowed for all Trace Logs (Mb)**, specify an approximate value for the most disk space that the trace file for this instance can occupy.

9 (Optional) To allow the Remote Loader to start automatically when the computer starts, select **Establish Remote Loader Service for this driver instance**.

NOTE: If the SSL connection fails due to `handshaketimeout` when Remote Loader establishes connection with Identity Manager engine then, update the default `handshaketimeout` variable to 10000 and restart both driver and remote loader.

---

10 (Optional) To allow the Remote Loader to use the secure protocol while connecting to the Identity Manager engine, specify the secure protocol version in the Remote Loader configuration file. For example: `secureprotocol=TLSv1_2`

For more information, see "Understanding the Configuration Parameters for the Remote Loader" on page 34.

NOTE: Skip this step if you already configured the secure protocol version on the driver.

---

11 (Optional) To allow the Remote Loader communication to use the protocols specified by Suite B, specify `enforceSuiteB=true` in the Remote Loader configuration file. This communication is supported only on TLS 1.2 protocol.

For more information, see "Understanding the Configuration Parameters for the Remote Loader" on page 34.

NOTE: Skip this step if you already enabled Suite B communication on the driver.

---

12 Click **OK**.

Modifying an Existing Driver Instance in the .NET Remote Loader

1. In the Remote Loader Console, select the driver instance from the **Description** column.
2. Click **Stop**.
3. Enter the password for the Remote Loader, then click **OK**.
4. Click **Edit**.
5. Modify the configuration information. For more information about each parameter, see "Creating a New Driver Instance in the .NET Remote Loader" on page 54.
6. To save the changes, click **OK**.

Configuring Identity Manager Drivers to Work with the Remote Loader

You can configure a new driver or enable an existing driver to communicate with the Remote Loader. You must set up an Identity Manager application shim for use with the Remote Loader.

NOTE: This section provides general information about configuring drivers so that they communicate with the Remote Loader. For driver-specific information, refer to the relevant driver implementation guide at the Identity Manager Driver documentation website (https://www.netiq.com/documentation/identity-manager-49-drivers/).

To add a new or modify an existing Driver object in either Designer or Identity Console, you must configure settings that enable the driver instance for the Remote Loader. For more information about the parameters used in this section, see "Understanding the Configuration Parameters for the Remote Loader" on page 34.

1. From **Overview**, select the Identity Manager Driver object.
2. In the properties of the Driver object, complete the following steps:
   - **2a** For **Driver Module**, select **Connect to Remote Loader**.
   - **2b** For **Driver Object Password**, specify the password that the Remote Loader uses to authenticate itself to the Identity Manager engine server.

     This password must match the password for the driver object defined in the Remote Loader.
   - **2c** For **Remote Loader Connection Parameters**, specify the information required to connect to the Remote Loader. Use the following syntax:

     `hostname=xxx.xxx.xxx.xxx port=xxxx kmo=certificatename localaddress=xxx.xxx.xxx.xxx`

     where

     hostname

     Specifies the IP address for the server that hosts the Remote Loader. For example, `hostname=192.168.0.1`.

     port

     Specifies the port that the Remote Loader listens on. The default is 8090.

     kmo

     Specifies the key name of the Key Material Object containing the keys and certificate used for SSL connections. For example, `kmo=remotecert`.

     localaddress

     Specifies the source IP address if more than one IP addresses are configured on the server that hosts the Identity Manager engine.
   - **2d** For **Remote Loader Password**, specify the password required for the Identity Manager engine (or Remote Loader shim) to authenticate to the Remote Loader.
3. Define a security-equivalent user.
4. Click **Next**, then click **Finish**.

Configuring Mutual Authentication with the Identity Manager Engine

You can configure mutual authentication to ensure secure communication between the Remote Loader and the Identity Manager engine. Mutual authentication uses certificates for handshake instead of passwords. The Remote Loader and the Identity Manager engine authenticate each other by exchanging and validating the public key certificate or digital certificate issued by the trusted Certificate Authorities (CAs) or self-signed certificates. If mutual authentication succeeds, the Remote Loader authenticates to the engine. Synchronization traffic occurs after both the Remote Loader and the Identity Manager engine establish trust that they are communicating with an authorized entity.

Perform the following tasks to configure mutual authentication:

- "Exporting Certificates for the Identity Manager Engine and the Remote Loader" on page 57
- "Enabling a Driver for Mutual Authentication" on page 60

Exporting Certificates for the Identity Manager Engine and the Remote Loader

For mutual authentication to work properly, you need a server certificate for the engine and a client certificate for the Remote Loader. You can export the certificates from eDirectory or import them from a third-party vendor. In most cases, you will export a server certificate from eDirectory with no further investment. In some cases, you might want to export a third-party client certificate for the Remote Loader.

- "Exporting a Certificate from eDirectory" on page 57
- "Exporting a Third-Party Certificate for Remote Loader" on page 60

Exporting a Certificate from eDirectory

A certificate object in the Identity Vault is called Key Material Object (KMO). This object securely contains both the certificate data including the public key and the private key associated with the certificate used for SSL connections. For mutual authentication, you need two KMOs, each for the engine and the Remote Loader.

You can export an existing KMO or create a new KMO and then export it. The process of creating a client KMO and a server KMO is different.

Creating KMOs

You must create a server KMO before creating a client KMO. To create a KMO, perform the following steps:

1. Log in to Identity Console
2. Click **Certificate Management > Server Certificate Management** options from the Identity Console landing page.
3. Click **+ Create Server Certificate**.
4. Specify the server to own the certificate that you created.
5. Specify a nickname for the certificate.

For example, `serverkmo` for server certificate and `clientkmo` for client certificate.

6 Select **Custom** in the certificate creation method, then click **Next**.

7 Keep the default **Organizational Certificate Authority** selection and click **Next**.

8 (Conditional) If you are creating client KMO.

   8a Select **Enable Extended key usage**.

   8b Select **Custom**, then select **User Authentication**.

   8c Click **Next**.

NOTE: For server KMO, keep the default selections and click **Next**.

---

9 Specify **Validity period** for the KMO.

   Ensure that Identity Console server time is synchronized with your Identity Manager components and the connected application.

10 Review the summary, click **OK**, then click **OK**.

11 Repeat these steps to create a client KMO.

Exporting KMOs

Export the KMOs from eDirectory that the engine and the Remote Loader will use for authenticating with each other.

To export the KMO for the Identity Manager engine, run the DirXML Command Line (dxcmd) utility:

```
dxcmd -user <admin DN> -password <password of admin> -exportcerts <kmoname>
<server|client> <java|native|dotnet> <output dir>
```

where

- `user` specifies the name of a user with administrative rights to the driver.
- `password` specifies the password of the user with administrative rights to the driver.
- `exportcerts` exports the certificates and private/public keys from eDirectory. You must specify whether you are exporting a server or a client certificate, the type of driver that will use the certificate, and a destination folder where the command will store this information.

Example1: `dxcmd -user admin.sa.system -password novell -exportcerts serverkmo server java 'C:\certs'`

This command generates the `serverkmo_server.ks` file in the `C:\certs` directory. The default keystore password and key password is `dirxml.`

Example 2: `dxcmd -user admin.sa.system -password novell -exportcerts serverkmo server java '/home/certs'`

This command generates the `serverkmo_server.ks` file in the `/home/certs/` directory. The default password for the keystore is `dirxml.`

When running the dxcmd command for exporting the KMO for the Remote Loader, the following considerations apply:

- The dxcmd utility runs in LDAP mode. When you use it for the first time, it prompts for specifying the choice of trusting the certificate from eDirectory. Depending on your environment, you can choose to trust the certificate only for the current session, for the current and future sessions, trust all certificates, or select not to trust the certificate.
- If the Remote Loader is running on the Identity Manager server, run the command in either LDAP or dot format. If the Remote Loader is installed on a separate server, run the command only in LDAP format.
- Specify the `-host` parameter in the command to resolve the server IP address or hostname to be able to authenticate with the Identity Manager server.

Run the command using the following syntax:

```
dxcmd -dnform ldap -host <IP address of the host> -user <admin DN> -
password <password of admin> -exportcerts <kmoname> <client>
<java|native|dotnet> <output dir>
```

Table 3-2 *Example commands for different types of drivers*

Type of Driver	Command	Output
Java Driver	`dxcmd -dnform ldap -host 194.99.90.218 -user cn=admin,ou=sa,o=system -password novell -exportcerts clientkmo client java 'C:\certs'`	`clientkmo_client.ks` file in the `C:\certs` directory The default password for the keystore is `dirxml.` The default **Private Key Password** is `dirxml`.
Native Driver	`dxcmd -dnform ldap -host 194.99.90.218 -user cn=admin,ou=sa,o=system -password novell -exportcerts clientkmo client native 'C:\certs'`	`clientkmo_clientcert.pem`, `clientkmo_clientkey.pem`, and `trustedcert.b64` files in the `C:\certs` directory. The default key password is `dirxml`.
.NET Driver	`dxcmd -dnform ldap -host 194.99.90.218 -user cn=admin,ou=sa,o=system -password novell -exportcerts clientkmo client dotnet 'C:\certs'`	`clientkmo_clientcert.pfx` and `trustedcert.b64` files in the `C:\certs` directory. The default key password of `clientkmo_clientcert.pfx` is `dirxml`.

Exporting a Third-Party Certificate for Remote Loader

To use third party certificates with the Remote Loader, you need to export a certificate in the `.pfx` file and a trusted root file in Base 64 format and then convert the `.pfx` certificate to the format that the driver uses. For example, a native driver requires the private key and the certificate key in `.pem` format while a Java driver requires the keystore in `.jks` format. The .NET driver uses the file in `.pfx` format. So you need not convert the file for a .NET driver.

Native driver

Complete the following steps:

1. Retrieve the private key in `.pem` format from the `.pfx` file.

   Enter a command, such as `openssl pkcs12 -in servercert.pfx -out serverkey.pem`

2. Retrieve the certificate key in `.pem` format from the `.pfx` file.

   Enter a command, such as `openssl pkcs12 -in servercert.pfx -out servercert.pem`

Java driver

Create a Java keystore from the `.pfx` file. Enter the following command:

```
keytool -importkeystore -srckeystore servercert.pfx -srcstoretype
pkcs12 -destkeystore servercert.jks -deststoretype JKS
```

This command prompts you to type source keystore password (`srckeystore passwd`) and destination keystore password (`dest keystorepasswd`). Enter these passwords appropriately.

As a final step, specify the information in the Remote Loader configuration file depending on the type of the driver. For more information, see Enabling a Driver for Mutual Authentication.

Enabling a Driver for Mutual Authentication

You enable a driver communication for mutual authentication by performing the following tasks:

- "Configuring a Driver Using KMO or Keystore" on page 60
- "Configuring the Remote Loader for Driver Instances" on page 63

Configuring a Driver Using KMO or Keystore

You can configure the driver by using KMO or keystore in Designer or Identity Console.

Designer

Before configuring a driver using KMO or keystore in Designer, ensure you have completed the basic driver configuration as follows:

1. Open your project in Designer.
2. In the palette of the **Modeler** view, select the driver that you want to create.
3. Drag the icon for the driver onto the **Modeler** view.
4. Follow the steps in the installation wizard.

5 In the Remote Loader window, select **yes**.

   5a **Host name:** Specify the hostname or IP address of the server where the driver's Remote Loader service is running. For example, enter **Host name** as `192.168.0.1`. If you do not specify a value for this parameter, the value defaults to `localhost`.

   5b **Port:** Specify the port number where the Remote Loader is installed and is running for this driver. The default port number is `8090`.

6 Click **Next**.

7 Follow the rest of the instructions in the wizard until you finish installing the driver.

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

To modify the driver configuration using KMO or Keystore

1 In the **Outline** view of Designer, right-click the driver.

2 Select **Properties**.

3 In the navigation pane, select **Driver Configuration**.

4 In **Authentication**, select **Enable Mutual Authentication** and specify the following parameters:

KMO

Specifies the name of the server KMO.

Other parameters

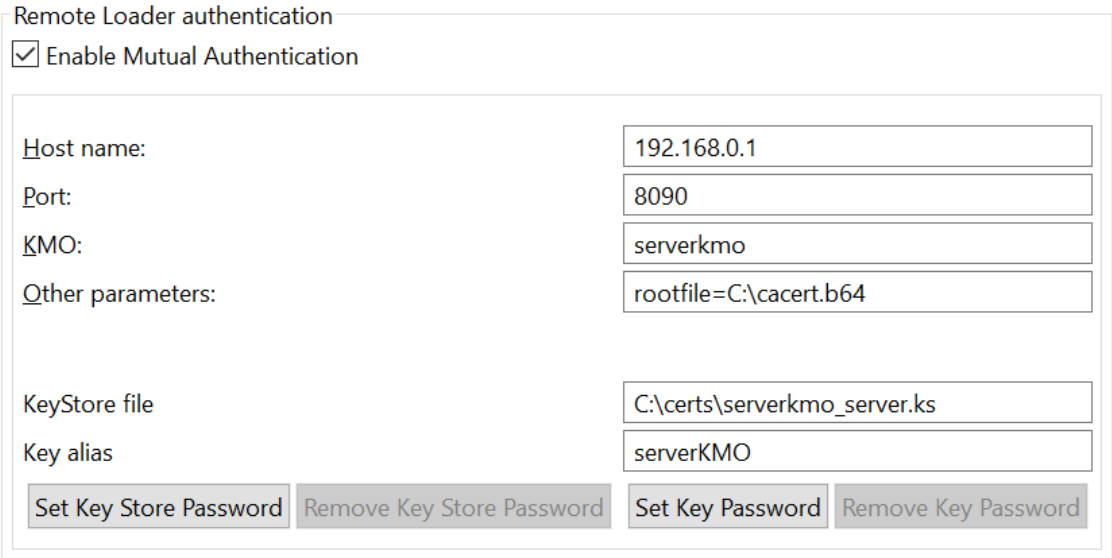
Specifies the `rootfile` and its absolute path.

KeyStore file

Specifies the absolute path of the keystore file.

Key alias

Specifies the name of the server KMO.



5 **Set Key Store Password**.

6 **Set Key Password**.

NOTE: By default, the **Key Store Password** and **Key Password** is set to `dirxml`.

You can also set Keystore and key password using `dxcmd` command.

```
dxcmd -user <administrative_user> -password <admin_password>
```

1. Select `Driver Operations`.
2. Select the driver that you want to set keystore and key password.
3. Select `Password Operations`.
4. Select `Set Keystore password for Mutual Authentication` and enter the Keystore password.
5. Select `Set Key password for Mutual Authentication` and enter the key password.

Identity Console

To modify the configuration in Identity Console:

1. Click the **Drivers** tab from the Identity Console home screen.
2. Click the driver name link for the driver that you want to configure.
3. Select **Connect to Remote Loader** under **Connection Parameters** tab.
4. (Windows) Specify the following connection details:

   4a In Remote loader connection parameters, specify the following connection details:

   ```
   KMO=<server_KMO_name>
   rootfile=<absolute path to the file>
   ```

   For example,

   ```
   KMO=serverkmo
   rootfile=C:\cacert.b64
   ```

   4b Set the **Application password**.

   4c Select **Enable Mutual Auth**.

   4d To use keystore method, specify the following:

   Key alias

   Specifies the name of server KMO and set Key password.

   For example: `serverKMO`

   Keystore file

   Specifies the absolute path of the keystore file and set the Keystore password.

   For example: `C:\certs\serverkmo_server.ks`

   4e Click **Save**.

   NOTE: When you enable mutual authentication, configuring **Remote Loader Password** and **Driver Object Password** are not necessary.

Configuring the Remote Loader for Driver Instances

You must configure the driver instance in the Remote Loader configuration file. Ensure that you specify the absolute path to the directory that stores the key file, certificate file, and the root file in the Remote Loader configuration file for a driver.

Adding a New Remote Loader Driver Instance on Windows

1. Right-click on the **Identity Manager Remote Loader Console** application and select **Run as administrator**.
2. Click **Add** to add a new Remote Loader instance.
3. Specify the **Description** and select the type of driver.
4. Specify the **Connection port** that is used to connect Remote Loader and Identity Manager engine.
5. Specify the **Command port** for your Remote Loader instance.
6. Select **Mutual Authentication** and specify the required type of driver:
   - **Java driver:** Browse the path of the keystore file that contains the certificate. The keystore file must contain at least one public/private key pair.

     Keystore File

     Specifies the path to the Java keystore file you want to use for authentication. The keystore file contains encryption keys and certificates. For example, `clientkmo_client.ks` in the `C:\certs\` directory created by `dxcmd` in "Exporting a Certificate from eDirectory" on page 57.

     Key Alias

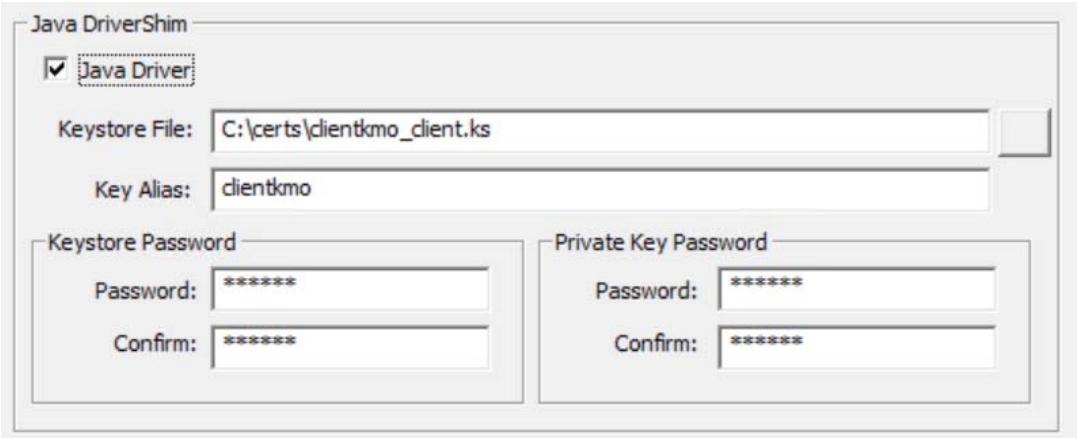
     Specifies the name of the public/private key pair in the keystore file that you want to use for symmetric key generation. For example, `clientkmo`.

     Keystore Password

     Specifies the password used to load the keystore file.

     Private Key Password

     Specifies the password for the private key stored in the keystore. Identity Manager uses this key for encrypting the SSL communication.



- **Native driver:** Browse the path to the key file that stores the certificate for authentication. The key file must be in Base 64 format.

  Key File

  Specifies the path to the file that stores the key for authentication. For example, `clientkmo_clientkey.pem` file in the `C:\certs\` directory created by `dxcmd`.

  Key Password

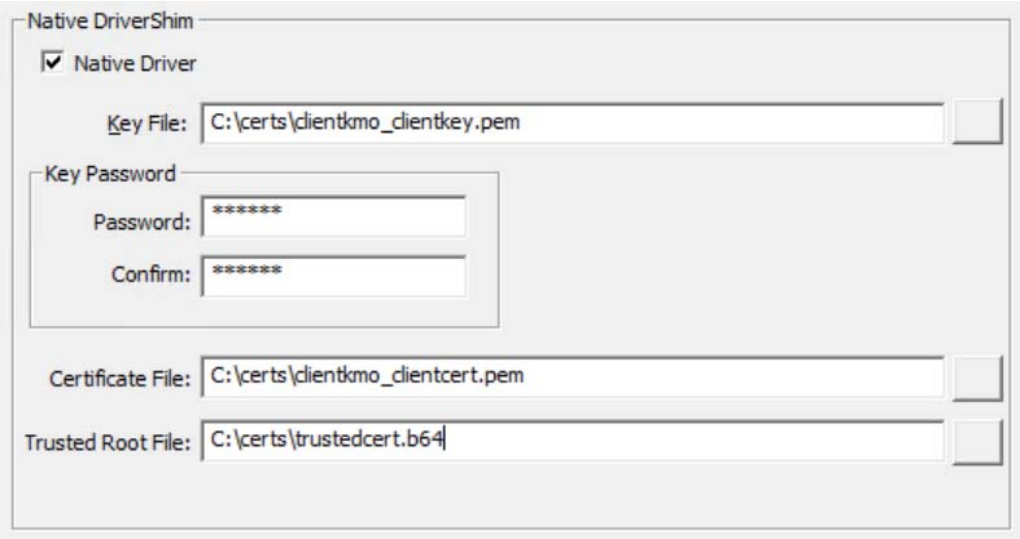
  Specifies the password for the private key used for authentication.

  Certificate File

  Specifies the file where the certificates are stored. The certificate file must be in Base 64 format. For example, `clientkmo_clientcert.pem` in the `C:\certs\` directory created by `dxcmd` in "Exporting a Certificate from eDirectory" on page 57.

  Trusted Root File

  Specifies the name of the file that contains the trusted root certificate of the issuer of the certificate that the remote interface shim uses. The trusted root file must be in Base 64 format. For example, `trustedcert.b64` files in the `C:\certs\` directory created by `dxcmd` in "Exporting a Certificate from eDirectory" on page 57.



- **.NET driver:** Browse the path to the key file that stores the certificate for authentication.

  Key File

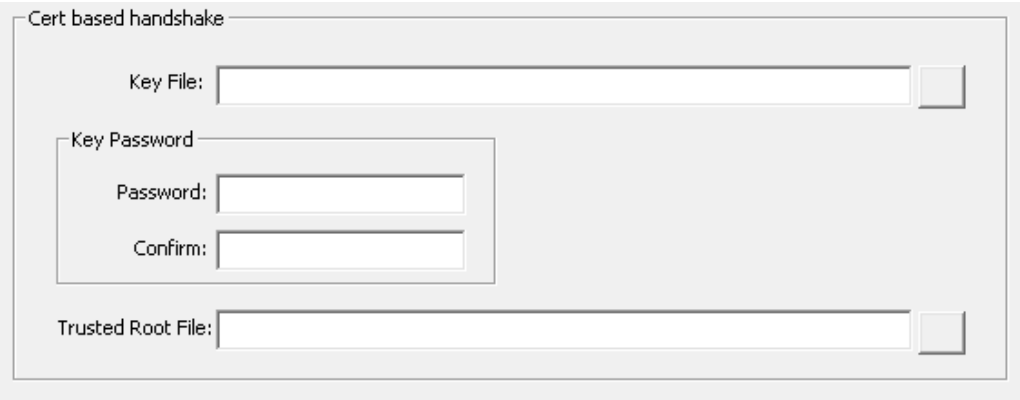
  Specifies the path to the file that stores the key for authentication. For example, `clientkmo_clientcert.pfx` in the in the `C:\certs\` directory created by `dxcmd` in "Exporting a Certificate from eDirectory" on page 57.

  Key Password

  Specifies the password for the private key used for authentication.

Trusted Root File

Specifies the name of the file that contains the trusted root certificate of the issuer of the certificate that the remote interface shim uses. The trusted root file must be in Base 64 format. For example, `trustedcert.b64` file in the `C:\certs\` directory created by `dxcmd` in "Exporting a Certificate from eDirectory" on page 57.



For more information about the output files generated for this driver using `dxcmd` tool, see Table 3-2, "Example commands for different types of drivers," on page 59.

Configuring the Remote Loader on Windows

You must configure the driver instance in the Remote Loader configuration file. Ensure that you specify the absolute path to the directory that stores the keystore file, key file, certificate file, and the root file in the Remote Loader configuration file for a driver.

1. In the Remote Loader Console, select the driver instance from **Description** column.
2. Click **Stop**.
3. Enter the password for the Remote Loader, then click **OK**.
4. Click **Edit** and perform Step 6 from "Adding a New Remote Loader Driver Instance on Windows" on page 63
5. Click **OK**.

Configuring the Remote Loader on UNIX or Linux

Modify the Remote Loader configuration file for a driver to include the content for enabling mutual authentication. The file is located in the `/opt/novell/dirxml/doc` directory.

To modify the configuration:

1. Log in to the server where you installed the driver and Remote Loader.
2. Stop the Remote Loader.

   For example, enter the following command:

   ```
   rdxml -config /home/drivershim.conf -u
   ```

3. Provide keystore or key password based on the Remote Loader type:

Java Remote Loader:

Specify the combination of keystore password and key password using the following syntax:

```
dirxml_jremote -config /home/drivershim.conf -ksp
<keystorepassword> -kp <keypassword>
```

For example,

```
dirxml_jremote -config /home/drivershim.conf -ksp dirxml -kp dirxml
```

Native Remote Loader:

Specify the keypassword using the following syntax:

```
dirxml_jremote -config /home/drivershim.conf -kp <keypassword>
```

For example,

```
dirxml_jremote -config /home/drivershim.conf -kp dirxml
```

4 In a text editor, open the Remote Loader configuration file for the driver.

5 Add the content required for enabling mutual authentication to the file.

- For example, for a Java driver, add this entry:

```
-connection "port=8090 useMutualAuth=true keystore='/home/certs/
clientkmo_client.ks' key='clientkmo'
```

- For example, for a Native driver, add this entry:

```
-connection "useMutualAuth=true port=8090 rootfile='/home/certs/
trustedcert.b64' certfile='/home/certs/clientkmo_clientcert.pem'
keyfile='/home/certs/clientkmo_clientkey.pem' certform=PEM
keyform=PEM"
```

6 Save and close the file.

7 Restart the driver.

Starting and Stopping the Remote Loader

The Remote Loader is either a service or a daemon, which occasionally must be restarted. This chapter explains how to stop and start the Remote Loader.

- "Starting a Driver Instance in the Remote Loader" on page 67
- "Stopping a Driver Instance in the Remote Loader" on page 69
- "Stopping, Starting, or Restarting a Driver in Designer" on page 70
- "Stopping, Starting, or Restarting a Driver in Identity Console" on page 70

Starting a Driver Instance in the Remote Loader

You can configure each platform to automatically start a driver instance when the host computer starts. You can also manually start an instance.

- "Starting Driver Instances on UNIX or Linux" on page 67
- "Starting Driver Instances on Windows" on page 68

Starting Driver Instances on UNIX or Linux

NetIQ provides two ways that you can start a driver instance for the Remote Loader on UNIX or Linux computers:

- "Starting Driver Instances Automatically on UNIX or Linux" on page 67
- "Using the Command Line to Start Driver Instances on UNIX or Linux" on page 67

Starting Driver Instances Automatically on UNIX or Linux

You can configure a driver instance for the Remote Loader to start automatically when the computer starts. Place your configuration file in the `/etc/opt/novell/dirxml/rdxml` directory.

Using the Command Line to Start Driver Instances on UNIX or Linux

For Linux platforms, the binary component `rdxml` supports command line functionality for the Remote Loader. This component is located by default in the `/usr/bin/` directory.

For more information about the parameters used in this section, see "Understanding the Configuration Parameters for the Remote Loader" on page 34.

1. Open a command prompt.
2. To specify the passwords for authenticating the driver instance to the Identity Manager engine, enter one of the following commands:
   - **Linux**: `rdxml -config` *`filename`* `-sp` *`password password`*
   - **UNIX**: `dirxml_jremote -config` *`config_file`* `-sp` *`password password`*
3. To start the driver instance, enter the following command:
   - **Linux**: `rdxml -config` *`filename`*
   - **UNIX**: `dirxml_jremote -config` *`filename`*
4. Log in to Identity Console, then start the driver.
5. Confirm that the Remote Loader is working properly.
   - **Linux**: Use the `ps` command or a trace file to determine whether the command and connection ports are listening.
   - **UNIX**: Monitor the Java Remote Loader by using the tail command on the tracefile:

     ```
     tail -f trace filename
     ```

If the last line of the log shows the following text, the loader is successfully running and awaiting connection from the Identity Manager remote interface shim:

```
TRACE: Remote Loader: Entering listener accept()
```

The Remote Loader loads the Identity Manager application shim only when the Remote Loader is in communication with the remote interface shim on the Identity Manager engine server. This means, for example, that the application shim shuts down if the Remote Loader loses communication with the server.

Starting Driver Instances on Windows

NetIQ provides three ways that you can start a driver instance for the Remote Loader on Windows computers:

- "Starting Driver Instances Automatically on Windows" on page 68
- "Using the Console to Start Driver Instances on Windows" on page 68
- "Using the Command Line to Start Driver Instances on Windows" on page 68

Starting Driver Instances Automatically on Windows

You can configure a driver instance for the Remote Loader to start automatically when the Windows computer starts.

1. Open the Remote Loader Console.

   During installation, if you created a shortcut for the Remote Loader Console, use the `Identity Manager Remote Loader Console` icon on the desktop. Otherwise, run the `rlconsole.exe` located by default in `C:\novell\remoteloader\`*`nn`*`bit`.
2. Select a driver instance, then click **Edit**.
3. Select **Establish a Remote Loader service for this driver instance**.
4. Save your changes, and then close the console.

Using the Console to Start Driver Instances on Windows

1. Open the Remote Loader Console.

   During installation, if you created a shortcut for the Remote Loader Console, use the `Identity Manager Remote Loader Console` icon on the desktop. Otherwise, run the `rlconsole.exe` located by default in `C:\novell\remoteloader\`*`nn`*`bit`.
2. Select a driver instance, then click **Start**.

Using the Command Line to Start Driver Instances on Windows

The `dirxml_remote.exe` file supports command line functionality for the Remote Loader. The executable is located by default in the `c:\novell\RemoteLoader` directory. For more information about the parameters used in this section, see "Understanding the Configuration Parameters for the Remote Loader" on page 34.

1. Open a command prompt.
2. To specify the passwords for authenticating the driver instance for the Remote Loader to the Identity Manager engine, enter the following command:

   ```
   dirxml_remote -config filename -setpasswords password password
   ```

For example:

```
dirxml_remote -config config.txt -sp Novell4 idmpwd6
```

3 To start the driver instance, enter the following command:

```
dirxml_remote -config filename
```

For example:

```
dirxml_remote -config config.txt
```

4 Log in to Identity Console, then start the driver.

5 Confirm that the Remote Loader is working properly.

The Remote Loader loads the Identity Manager application shim only when the Remote Loader is in communication with the remote interface shim on the Identity Manager engine server. This means, for example, that the application shim shuts down if the Remote Loader loses communication with the server.

6 (Conditional) If you did not previously install the Remote Loader as a Win32 service, enter the following command:

```
dirxml_remote -config filename -service install
```

For example:

```
dirxml_remote -config config.txt -service install
```

Stopping a Driver Instance in the Remote Loader

Each platform has a different method for stopping a driver instance in the Remote Loader. For more information about the parameters used in this section, see "Understanding the Configuration Parameters for the Remote Loader" on page 34.

NOTE: ◆If you run multiple instances of the Remote Loader on a UNIX or Linux computer, include the `-cp` *command port* option to ensure that the Remote Loader can stop the appropriate instance.

- When you stop a driver instance, you must have sufficient rights or specify the Remote Loader password. For example, the Remote Loader is running as a Windows service. You have sufficient rights to stop it. You enter a password, but realize that it is incorrect. The Remote Loader stops anyway, because the Remote Loader does not actually "accept" the password. Instead, it ignores the password because the password is redundant in this case. If you run the Remote Loader as an application rather than as a service, the password is used.

To stop a driver instance:

Linux

Enter the `rdxml -config filename -u` command. For example:

```
rdxml -config config.txt -u
```

UNIX

Enter the `dirxml_jremote -config filename -u` command. For example:

```
dirxml_jremote -config config.txt -u
```

Windows

Use the Remote Loader Console.

During installation, if you created a shortcut for the Remote Loader Console, use the `Identity Manager Remote Loader Console` icon on the desktop. Otherwise, run the `rlconsole.exe` located by default in `C:\novell\remoteloader\`*nn*`bit`.

Stopping, Starting, or Restarting a Driver in Designer

1. Open a project in the Modeler, then right-click the driver line.
2. Click **Live**, then click the appropriate option to stop, start, or restart the driver.

Stopping, Starting, or Restarting a Driver in Identity Console

1. Click the **IDM Administration** tab from the Identity Console home screen.
2. Click the specific driver set object on the right hand side of your computer screen to display all the drivers associated to it.
3. Click the Action menu icon on the specific driver whose status you want to change, then click the appropriate option to stop, start, or restart the driver.

Verifying the Configuration

For more information about starting and stopping the Remote Loader, see "Starting and Stopping the Remote Loader" on page 66.

1. Start the driver using Identity Console.
2. Manage Remote Loader using any of the following ways:

   Remote Loader user interface

   1. Right-click the **Identity Manager Remote Loader** console and select **Run as administrator**.
   2. You can **Start**, **Stop**, **Add**, **Remove**, and more operations using Remote Loader interface.

   NOTE: To run Remote Loader as a service, select **Establish a Remote Loader service for this driver instance**. De-selecting this option runs Remote Loader as an application.

   Remote Loader console

   Navigate to the Remote Loader installed location and run the following commands in the command prompt:

   1. To start or load the Remote Loader instance:

      For Java Remote Loader:

      ```
      dirxml_jremote -config <configuration_filename> -ksp
      <keystore_password> -kp <keypassword>
      ```

      ```
      dirxml_jremote -config <configuration_filename>
      ```

      For Native Remote Loader:

```
dirxml_remote -config <configuration_filename> -ksp
<keystore_password> -kp <keypassword>
```

```
dirxml_remote -config <configuration_filename>
```

For .Net Remote Loader:

```
RemoteLoader.exe  -config <configuration_filename> -ksp
<keystore_password> -kp <keypassword>
```

```
RemoteLoader.exe  -config <configuration_filename>
```

2. To stop or unload the Remote Loader instance, append -u at the end of the previous command. For example

   For Java Remote Loader:

```
dirxml_jremote -config <configuration_filename> -u
```

   For Native Remote Loader:

```
dirxml_remote -config <configuration_filename> -u
```

   For .Net Remote Loader:

```
RemoteLoader.exe  -config <configuration_filename> -u
```

NOTE: To run the Remote Loader instance as a service use the following command:

```
dirxml_remote -config config.txt -service install
```

---

4 Viewing Version Information

The Identity Manager engine, the driver shims, and the driver configuration files each contain a separate version number. The Version Discovery option in Identity Console helps you find the versions of the Identity Manager engine and the driver shims versions. The driver configuration files contain their own naming convention. To view the version information:

1. In Identity Console, select **IDM Administration**, then click the Action menu (three dots) of the appropriate **Driver Set > Driver Set Properties > Inspector and Statistics > Version Discovery**.
2. View a top-level display of versioning information.
   - The eDirectory tree that you are authenticated to
   - The driver set that you selected
   - Servers that are associated with the driver set

     If the driver set is associated with two or more servers, you can view Identity Manager information about each server.
   - Drivers
3. Click the **View** icon to display a textual representation of the same information contained in the top-level view.
4. Click the **Export** button to export and save the text to a file on your local or network drive.

5 Backing Up Drivers

After you have created a driver, it is important to create a backup of the driver. You can use Designer 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 purged. When the XML file is imported again, new associations are 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. See "Documenting Projects" in the *NetIQ Designer for Identity Manager Administration Guide*.

You can also run the driver in factory mode, if you created the driver with packages.

Exporting the Driver in Designer

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

Running the Driver in Factory Mode

If you created the driver in Designer using packages, you can run a driver in the default factory mode.

There are two options for using Factory mode:

- **Strict:** Designer removes all customizations and custom configurations from your driver. Custom configurations are new policies, jobs, mapping policies, or other objects created on the driver.
- **Relaxed:** Designer removes all customizations but no custom configurations from your driver.

To run the driver in the factory mode:

1. In Designer, right-click the driver, then click **Driver > Properties**.
2. Click **Packages**, then select **Run driver in Factory mode**.
3. Select how Package Manager handles the customizations and custom configuration of your driver. You can select either **Strict** or **Relaxed**.
4. Click **Activate** to save the selected change.

5 (Optional) Click the **Configure Factory mode** icon if you want to change the selected option, then click **Activate** again.

6 Click **Apply** or **OK** to make the change active.

For more information, see "Running a Driver in Factory Mode" in the *NetIQ Designer for Identity Manager Administration Guide*.

6 Monitoring Driver Health

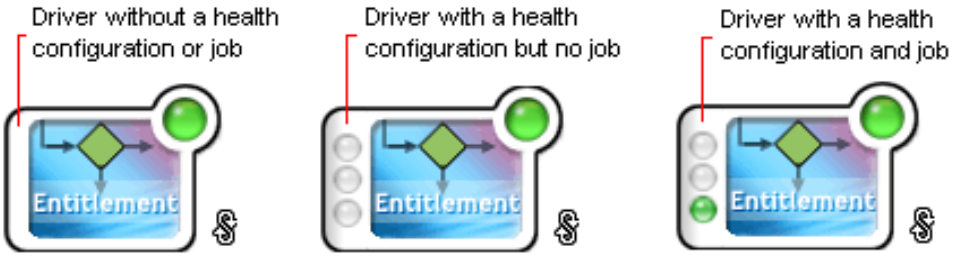
Driver health monitoring allows you to view a driver's current state of health as green, yellow, or red, and to define the actions to perform in response to each of these health states.

You create the conditions (criteria) that determine each of the health states, and you also define the actions you want performed whenever the driver's health state changes. For example, if the driver's health changes from a green state to a yellow state, you can perform such actions as restarting the driver, shutting down the driver, and sending an email to the person designated to resolve issues with the driver.

You can also define custom states. Whenever the conditions for the custom state are met, the associated actions are performed regardless of the driver's current state of green, yellow, or red.

The driver's health state is not monitored unless both a health configuration and a health job exist and the health job is running. (The health configuration for drivers is automatically created.) If the configuration and job exist and the job is running, the driver icon displays a green, yellow, or red indicator. Otherwise, the indicator does not appear or appears without a color.

Figure 6-1 *Driver health indicator*



To turn on health monitoring for the driver, create a driver health job. After you have created the driver's health job, you can use the steps in the following sections to modify the conditions and actions associated with each health state and to create one or more custom states:

- "Modifying the Conditions for a Driver Health Configuration" on page 79
- "Modifying the Actions for a Driver Health Configuration" on page 82
- "Creating a Custom State" on page 84

Creating a Driver Health Job

The health of a driver is evaluated during the periodic execution of a Driver Health job. The job evaluates the conditions for the health states and assigns the driver the appropriate state. The job also executes any actions associated with the assigned state.

If a Driver Health job does not exist, the Driver Health Configuration page displays a **Run the New Driver wizard and import the Driver Health Job's configuration** prompt. If the page does not display this prompt, the Driver Health job already exists, and you can skip to "Modifying the Conditions for a Driver Health Configuration" on page 79.

To create a Driver Health job:

1. In Identity Console, select the **Jobs** tab.
2. Click **New** to create a Driver Health Job.
3. Select the type as Health Job and ensure that the server on which the job will run on is selected.
4. Click **OK**.

After the job is created, you can adjust the job settings as desired. For example, you can modify how often the job runs, which drivers use the job, and how much data the job maintains to support transaction history. For instructions, continue with "Modifying the Driver Health Job's Settings" on page 78.

Modifying the Driver Health Job's Settings

The Driver Health job evaluates the conditions for the health states and assigns the driver the appropriate state. The job also executes any actions associated with the assigned state.

As with all driver jobs, there are several Driver Health job settings that you can modify to optimize health monitoring performance for your environment, including settings for how often the job runs, which drivers use the job, and how much data the job maintains to support transaction history.

To modify the job settings:

1. In Identity Console, select the **Jobs** tab.
2. Click the job you want to modify.
3. Open the Driver Health Configuration page for the driver that uses the Driver Health job you want to modify:
   - **3a** Open the Identity Manager Administration page.
   - **3b** In the **Administration** list, click **Driver Health Configuration**.
4. Click the Driver Health job.
5. Change the desired settings on the following tabs:
   - **Schedule:** The Driver Health job is a continuously running job, meaning that it does not stop unless a health state action shuts it down or it is shut down manually. The job must run continuously to be able to support transaction data collection for use in Transactions History conditions.

     If the job does stop, it is restarted based on the schedule. The default schedule checks every minute to see if the job is running. If the job is not running, it is started.

- **Scope:** By default, the job applies to all drivers in the driver set. This means that you need only one Driver Health job per driver set. However, you can create multiple Driver Health jobs for different drivers within the same driver set. For example, you might have some drivers whose health you want updated more frequently than other drivers, in which case you would need at least two Driver Health jobs.
- **Parameters:** You can change any of the following parameters:
  - **Login ID:** This defaults to the login ID that was used when creating the driver job. You should change this only if you want the driver to authenticate with different credentials.

    You need the following rights to run the health job:
    - Read permission with inheritance to the `DirXML-AccessConfigure` attribute of the Driver Set object
    - Read permission with inheritance to the `DirXML-AccessRun` attribute of the Driver Set object
    - Write permission with inheritance to the `DirXML-AccessSubmitCommand` attribute of the Driver Set object
  - **Login password:** This is the password required for the login ID that you supplied in the Login ID field.
  - **Subscriber Heartbeat:** Controls whether the Driver Health job does a heartbeat query on a driver's Subscriber channel before performing a health check on the driver.
  - **Polling interval:** Determines how often the job evaluates the conditions for the health states, assigns the driver the appropriate state, executes any actions associated with the assigned state, and stores the driver's transaction data. The default polling interval is one minute.
  - **Polling interval units:** Specifies the time unit (minutes, hours, days, weeks) for the number specified in the **Polling interval** setting.
  - **Duration sampling data is kept:** Specifies how long a driver's transaction data is kept. The default, two weeks, causes a transaction to be retained for two weeks before being deleted. A longer duration provides a greater time period that can be used in "Transactions History:" on page 80conditions, but requires more memory. For example, to use a Transactions History condition that evaluates of the number of publisher reported events for the last 10 days, you need to keep transaction data for at least 10 days.
  - **Duration units:** Specifies the time unit (minutes, hours, days, weeks) for the number specified in the Duration transaction data is kept setting.

6 Click **OK** to save your changes.

Modifying the Conditions for a Driver Health Configuration

You control the conditions that determine each health state. The green state is intended to represent a healthy driver, and a red state is intended to represent an unhealthy driver.

The conditions for the green state are evaluated first. If the driver fails to meet the green conditions, the yellow conditions are evaluated. If the driver fails to meet the yellow conditions, the driver is automatically assigned a red health state.

To modify the conditions for a state:

1. In Identity Console, open the Driver Health Configuration page for a driver whose conditions you want to modify:
   - **1a** Open the Identity Manager Administration page.
   - **1b** In the **Administration** list, click **Driver Health Configuration**.
2. Click the tab for the state (Green or Yellow) you want to modify.

   The tab displays the current conditions for the health state. Conditions are organized into groups, and logical operators, either AND or OR, are used to combine each condition and each group. Consider the following example for the green state:

   ```
   GROUP1
   Condition1 and
   Condition2
   Or
   GROUP2
   Condition1 and
   Condition2 and
   Condition3
   ```

   In the example, the driver is assigned a green state if either the GROUP1 conditions or the GROUP2 conditions evaluate as true. If neither group of conditions is true, then the conditions for the yellow state are evaluated.

   The conditions that can be evaluated are:

   - **Driver State:** Running, stopped, starting, not running, or shutting down. For example, one of the default conditions for the green health state is that the driver is running.
   - **Driver in Cache Overflow:** The state of the cache used for holding driver transactions. If the driver is in cache overflow, all available cache has been used. For example, the default condition for the green health state is that the Driver in Cache Overflow condition is false and the default for the yellow health state is that the Driver in Cache Overflow condition is true.
   - **Newest:** The age of the newest transaction in the cache.
   - **Oldest:** The age of the oldest transaction in the cache.
   - **Total Size:** The size of the cache.
   - **Unprocessed Size:** The size of all unprocessed transactions in the cache.
   - **Unprocessed Transactions:** The number of unprocessed transactions in the cache. You can specify all transactions types or specific transaction types (such as adds, removes, or renames).
   - **Transactions History:** The number of transactions processed at various points in the Subscriber or Publisher channel over a given period of time. This condition uses multiple elements in the following format:

     <*transaction type*> <*transaction location and time period* > <*relational operator*> <*transaction number*>.

     - <*transaction type*>: Specifies the type of transaction being evaluated. This can be all transactions, adds, removes, renames, and so forth.

- <*transaction location and time period*>: Specifies the place in the Subscriber or Publisher channel and the time period being evaluated. For example, you might evaluate the total number of transactions processed as Publisher reported events over the last 48 hours. By default, transaction history data is kept for two weeks, which means that you cannot specify a time period greater than two weeks unless you change the default Transaction Data Duration setting. This setting is specified on the Driver Health job. See "Modifying the Driver Health Job's Settings" on page 78 for information about changing the setting.
- <*relational operator*>: Specifies that the identified transactions must be equal to, not equal to, less than, less than or equal to, greater than, or greater than or equal to the <transaction number>.
- <*transaction number*>: Specifies the number of transactions being used in the evaluation.

The following provides an example of a Transactions History condition:

```
<number of adds> <as publisher commands> <over the last 10 minutes>
<is less than> <1000>
```

- **Available History:** The amount of transaction history data that is available for evaluation. The primary purpose for this condition is to ensure that a Transactions History condition does not cause the current state to fail because it does not have enough transaction history data collected for the time period being evaluated.

  For example, assume that you want to use the Transactions History condition to evaluate the number of adds as Publisher commands over the last 48 hours (the example shown in the Transactions History section above). However, you don't want the condition to fail if there is not yet 48 hours worth of data, which can be the case after the initial setup of the driver's health configuration or if the driver's server restarts (because transaction history data is kept in memory). Therefore, you create condition groups similar to the following:

  ```
  Group1 Available History <is less than> <48 hours> or Group2
  Available History <is greater than or equal to> <48 hours> and
  Transactions History <number of adds> <as publisher commands> <over
  the last 48 hours> <is less than> <1000>
  ```

  The state evaluates to true if either condition group is true, meaning that a) there is less than 48 hours of data, or b) there is at least 48 hours of data and the number of adds as Publisher commands over the last 48 hours is less than 1000.

  The state evaluates to false if both conditions evaluate to false, meaning that a) there is at least 48 hours of data and b) the number of adds as publisher commands over the last 48 hours is greater than 1000.

3 Modify the criteria as desired.

- To add a new group, click **New Group**.
- To add a condition, click the plus (+) icon next to the group heading.
- To reorder condition groups or individual conditions, select the check box next to the group or condition you want to move, then click the arrow buttons to move it up and down. You can also use the arrow buttons to move a condition from one group to another.
- To copy condition groups or individual conditions, select the check box next to the group or condition you want to copy, click **Edit > Copy selections to clipboard**, click the tab for the health state where you want to copy the group or condition, then click **Edit > Append items on clipboard**. For example, assume that you want to copy a condition from one condition

group to another. You would select the condition, copy it to the clipboard, then append it. The condition is added as its own condition group; if desired, use the arrow buttons to move it into another condition group.

- To move condition groups or individual conditions, select the check box next to the group or condition you want to move, click **Edit > Cut selections to clipboard**, click the tab for the health state where you want to move the group or condition, then click **Edit > Append items on clipboard**. For example, assume that you want to move a condition group from the green health state to the yellow health state. You would select the condition group, cut it to the Clipboard, open the yellow health state, then append it.

4 Click **Apply** to save your changes.

5 If you want to change the actions associated with the conditions you have set, continue with "Modifying the Actions for a Driver Health Configuration" on page 82.

Modifying the Actions for a Driver Health Configuration

You can determine the actions that you want performed when the driver health state changes. For example, if the state changes from green to yellow, you can shut down or restart the driver, generate an event, or start a workflow. Or, if the state changes from yellow to green, any actions associated with the green state are performed.

A health state's actions are performed only once each time the conditions are met; as long as the state remains true, the actions are not repeated. If the state changes because its conditions are no longer met, the actions are performed again the next time the conditions are met.

1 In Identity Console, open the Driver Health Configuration page for a driver whose actions you want to modify:

   1a Open the Identity Manager Administration page.

   1b In the **Administration** list, click **Driver Health Configuration**.

2 Click the **Green**, **Yellow**, or **Red** tab for the state whose actions you want to modify.

3 Click the plus (+) icon next to the **Actions** heading to add an action, then select the type of action you want:

   - **Start Driver:** Starts the driver.
   - **Stop Driver:** Stops the driver.
   - **Restart Driver:** Stops and then starts the driver.
   - **Clear Driver Cache:** Removes all transactions, including unprocessed transactions, from the cache.
   - **Send Email:** Sends an email to one or more recipients. The template you want to use in the email message body must already exist. To include the driver name, server name, and current health state information in the email, add the `$Driver$`, `$Server$`, and `$HealthState$` tokens to the email template and then include the tokens in the message text. For example:

     ```
     The current health state of the $Driver$ driver running on $Server$
     is $HealthState$.
     ```

IMPORTANT: To send emails to multiple users, separate each email address only with a comma (,). Do not use semi-colon instead of comma.

- **Write Trace Message:** Writes a message to the Driver Health job's log file or the driver set's log file if the trace file is not configured on the Driver Health job.
- **Generate Event:** Generates an event that can be used by Audit and Sentinel.
- **Execute ECMAScript:** Executes an existing ECMAScript. Use the or buttons to select the DirXML-Resource object that contains the ECMAScript.

  For information about how to construct ECMA scripts, refer to "Using ECMAScript in Policies" in the *NetIQ Identity Manager - Using Designer to Create Policies*.
- **Start Workflow:** Starts a provisioning workflow.
- **On Error:** If an action fails, instructs what to do with the remaining actions, the current health state, and the Driver Health job.
  - **Affect actions by:** You can continue to execute the remaining actions, stop execution of the remaining actions, or default to the current setting. The current setting applies only if you have multiple On Error actions and you set the Affect actions by option in one of the preceding On Error actions.
  - **Affect state by:** You can save the current state, reject the current state, or default to the current setting. Saving the state causes the state's conditions to continue to evaluate as true. Rejecting the state causes the state's conditions to evaluate as false. The current setting applies only if you have multiple On Error actions and you set the Affect state by option in one of the preceding On Error actions.
  - **Affect Driver Health Job by:** You can continue to run the job, abort and disable the job, or default to the current setting. Continuing to run the job causes the job to finish evaluating the conditions to determine the driver's health state and perform any actions associated with the state. Aborting and disabling the job stops the job's current activity and shuts down the job; the job does not run again until you enable it. The current setting applies only if you have multiple On Error actions and you set the Affect Driver Health Job by setting in one of the preceding On Error actions.

4. If you want the actions executed every time the conditions evaluate to true, click **Always execute actions when conditions are true**.

   By default, actions are performed only once while a driver's health state remains the same. Regardless of the number of times the conditions are evaluated, as long as the health state remains true, the actions are not repeated. For example, when the driver's health state changes from red to green, the green state's actions are executed. The next time the conditions are evaluated, if the health state is still green, the actions are not repeated.

   Selecting the **Always execute actions when conditions are true** setting causes the actions to be repeated each time the condition evaluates to true. For example, if the driver's health state repeatedly evaluates to green without changing to another state, the green state's actions are repeated after each evaluation.

5. Click **Apply** to save your changes.

Creating a Custom State

You can create one or more custom states to perform actions independent of the driver's current health state (green, yellow, red). If a custom state's conditions are met, its actions are performed regardless of the current health state.

As with the green, yellow, and red health states, a custom state's actions are performed only once each time the conditions are met; as long as the state remains true, the actions are not repeated. If the state changes because its conditions are no longer met, the actions are performed again the next time the conditions are met.

1. In Identity Console, open the Driver Health Configuration page for a driver for which you want to create a custom state:
   - **1a** Open the Identity Manager Administration page.
   - **1b** In the **Administration** list, click **Driver Health Configuration**.
2. On any of the tabs, click **Actions**, then click **New Custom State**.
3. Follow the instructions in "Modifying the Conditions for a Driver Health Configuration" on page 79 and "Modifying the Actions for a Driver Health Configuration" on page 82 to define the custom state's conditions and actions.

Memory Requirements for Driver Health

The combination of interval, interval-units, duration, and duration-units define how much sampling data is maintained by the Driver Health Job. The values for these parameters directly affect how much memory the Driver Health Job requires to run.

The number of samples per driver per server is calculated as:

Number of samples = ((duration * duration units) / (polling interval * polling units)) + 1

For example, if

duration = 12 hours
polling interval = 1 minute

Number of samples = (12*60) / (1*1) + 1 = 721

If there are 4 drivers on 1 server, total number of samples = 4*1*721 = 2884.

Each sample stores data from 5 points in the publisher channel and 5 points in the subscriber channel.

Publisher Channel Points:

publisher-commands
publisher-command-results
publisher-post-event-transformation
publisher-post-input-transformation
publisher-reported-events

Subscriber Channel Points:

subscriber-commands

subscriber-command-results

subscriber-pre-output-transformation

subscriber-post-event-transformation

subscriber-reported-events

A sample contains a list of IDs and counts for each point. IDs correspond to operations such as query, status, instance, add-association, and so on.

Consider the following driver cache statistics:

```
<subscriber>
   <operations>
        <command-results>
            <status>12</status>
            <instance>12</instance>
        </command-results>
   </operations>
</subscriber>
```

For subscriber-command-results, the list has IDs 7,21 (for instance and status) and counts 12,12.

Each sample consumes ~700 bytes.

721 samples consume ~ 500 KB. This is the memory requirement per driver.

For 4 drivers, 2 MB is required for storing sampling data.

7 Viewing Driver Statistics

You can use NetIQ Identity Console to view a variety of statistics for a single driver or for an entire driver set. This includes statistics such as the cache file size, the size of the unprocessed transactions in the cache file, the oldest and newest transactions, and the total number of unprocessed transactions by category (add, remove, modify, and so forth).

Viewing Statistics for an Individual Driver

1. Click **Identity Manager Overview** to display the Identity Manager Overview page.

   You use the Identity Manager Overview page to locate the driver set in which the driver resides.

2. In the **Search in** field, specify the fully distinguished name of the container where you want to start searching for the driver set, then click . Or, click the browse icon to browse for and select the container in the tree structure.

   Identity Console keeps a record of the objects you have previously selected, so you can also use the to select the container from a list of previously selected objects. Or, you can search from the root of the tree by clicking .

3. After the search completes and displays the driver sets, click the driver set in which the driver resides to display the Driver Set Overview page.

4. Locate the driver whose statistics you want to check, click the driver's **Status** icon (the green or red circle on the driver icon), then click **Statistics**.

Viewing Statistics for a Driver Set

1. Click **Driver Set Dashboard** to display the Driver Set Query page.

2. In the **Driver Set** field, specify the fully distinguished name of the driver set, then click **OK**. Or, click the browse icon to browse for and select the driver set in the tree structure, then click **OK**.

   Identity Console keeps a record of the objects you have previously selected, so you can also use the icon to select the container from a list of previously selected objects.

   A page appears that allows you to view the statistics for all of the drivers contained in the driver set.

   - To refresh the statistics, click **Refresh**, then select **Refresh now** or select a refresh interval.
   - To close the statistics for a driver, click the button in the upper right corner of the driver's statistics window.
   - To open the statistics for all drivers, click **Actions** > **Show all drivers**.
   - To collapse the list of unprocessed transactions for a driver, click the button located above the list. To collapse the list of unprocessed transactions for all drivers, click **Actions** > **Collapse all transactions**.

- To expand the list of transactions, click the button. To expand the list of unprocessed transactions for all drivers, click **Actions** > **Expand all transactions**.
- To change the layout of the driver dashboard, click **Actions**, then select a column layout.

8 Managing Associations between Drivers and Objects

Associations

A relationship is established between an Identity Vault object and a connected system object when the two objects represent the same entity. This relationship is called an association and is stored in the Identity Vault on the associated Identity Vault object. Identity Manager uses the association to keep track of which object in the connected system matches with an object in the Identity Vault. In almost all cases, this should be a 1:1 match to say that "Herman Munster, employee number 1234567" in the HR system matches exactly with the user object "hmunster13" in the Identity Vault, with "Munster, Herman" in Active Directory, and "hmunster13@example.com" in the email system.

Associations are stored only in the Identity Vault. The shim provides a unique key value for each application object and the Identity Manager engine manages the storage of those key values in the Identity Vault. On the Subscriber channel, the Identity Manager engine uses this value to allow the shim to modify the correct object in the connected system. On the Publisher channel, the shim supplies the association value, allowing the identity Manager engine to quickly and easily find the correct object in the Identity Vault to work with. The following Association states are stored in the Identity Vault:

- **0 Disabled:** Changes in the driver objects are not synchronized with the Identity Vault.
- **1 Processed:** A successful association has been created between driver objects and the Identity Vault.
- **2 Pending:** The Identity Manager engine identified a modification to an object, and attempted to match it or create it in the connected system, but was unable to do so.
- **3 Manual:** A manual association was created by the user.
- **4 Migrate:** The account was synchronized or migrated.
- **blank No association:** No association has been created.

No-Reference Associations

Identity Manager maintains associations in an eDirectory attribute (Syntax : SYN_PATH) named DirXML-Associations. This attribute has three parts to it.

```
dirxml-associations: cn=DT-1,cn=DS1,o=n#1#abc@novell.com
```

The first part is a `driver-dn`, which denotes the driver this association is for; the second field denotes the association state; and the final field denotes the association value. The part that is used to store the `driver-dn` is stored as an eDirectory DN. If there are any object renames or moves, the associations do not get broken and are preserved.

However, any updates to the referred object also result in a reference check. This causes a small overhead that can impact performance in very large deployments.

To improve performance in large deployments, a new no-reference association has been introduced in Identity Manager. Though the existing association continues to be the default option, Identity Manager provides you an option to switch to the new association format for a driver. In your Identity Manager deployment, some drivers can have the legacy reference association while others create a no-reference association. The driver's DN is maintained as a string with the new no-reference association. If you change this, the mapping of the object from the Identity Vault to the connected system might get broken.

A new attribute, `DirXML-AssociationsLite` of type `SYN_CI_STRING`, is included to store the no-reference association. The new attribute contains the stringized version of the object association.

```
dirxml-associationslite: \ABC-SLES10SP2X86-NDSTREE\n\DS1\bedir-174-18-4-
32#648F713EC4AB284967AB648F713EC4AB#1
```

The new association attribute uses "#" to delimit the components of the association. The first component is the complete `driver-dn` including the eDirectory tree name in the slash format. The second component is the association value and the last component is the association state.

A new attribute, `DirXML-UseNoRefAssoc` of type `SYN_BOOLEAN`, is included with the drivers to denote the type of association to be used for the drivers. The absence of this attribute or a value of false implies that the driver uses the legacy association attribute (`DirXML-Associations`). If the value is set to true, the driver uses the new association attribute (`DirXML-AssociationsLite`) for the association.

NOTE: You should use the new association format with Identity Manager Standard Edition that provides limited Reporting features or in very large deployments where referential checks cause considerable performance impact. If you use it with Identity Manager Advanced Edition, all aspects of the Reporting functionality might not work as expected.

Migration Between Associations

The dxcmd menu is updated to provide new actions to enable easier and seamless migration of association from one format to the other. Additionally, maintenance actions are available to handle the associations.

NOTE: The Association actions in dxcmd utility are hidden by default. You must invoke this utility by using the `-u` option to see the Association actions.

The association actions are available at two levels in the dxcmd menu. The association actions are available in the main dxcmd menu and also in the driver menu. You need to shut down the driver before performing these actions.

Association Actions in the Main Menu

You can select an association operation from the following list:

- 1: Migrate to no-ref association

  This action modifies the driver and sets the value of `DirXML-UseNoRefAssoc` to True. It also converts the existing object associations for this driver to use the `DirXMLAssociationsLite` attribute (new no-ref association).

- 2: Migrate to ref association

  This action modifies the driver and sets the value of `DirXML-UseNoRefAssoc` to False. It also converts the existing object associations for this driver to use the `DirXMLAssociations` attribute (legacy association).

- 3: Get no-ref associated entries

  Applicable only if using no-reference association for the driver. This action lists the object's associations.

- 4: Delete no-ref associated entries

  Applicable only if using no-reference association for the driver. This action deletes the object's associations.

- 5: Rename no-ref association

  Applicable only if using no-reference association for the driver. This action renames the object's associations. As the DN is stored as a string, renaming the driver or renaming, or moving the object (if using DN as association value) may break the association. You can use the rename action to correct this behavior.

- 99: Exit

Association Actions in the Driver Menu

You can select an association operation from the following list:

- 1: Migrate to no-ref association

  This action modifies the driver and sets the value of `DirXML-UseNoRefAssoc` to True. It also converts the existing object associations for this driver to use the `DirXMLAssociationsLite` attribute (new no-ref association).

- 2: Migrate to ref association

  This action modifies the driver and sets the value of `DirXML-UseNoRefAssoc` to False. It also converts the existing object associations for this driver to use the `DirXMLAssociations` attribute (legacy association).

- 3: Get to no-ref associated entries

  Applicable only if using no-reference association for the driver. This action lists the object's associations.

- 4: Delete no-ref associated entries

  Applicable only if using no-reference association for the driver. This action deletes the object's associations.

- 99: Exit

Tools for Managing Associations

NetIQ Identity Console provides two tools to enable you to view and manage the associations between drivers and objects (data):

- The **Driver Inspector** displays all objects associated with a driver and lets you perform various actions on those associations, such as deleting an object or modifying its properties.
- The **Object Inspector** displays all connected systems associated with an object. For each association, you can perform various actions, including viewing the object's data flow between the Identity Vault and the connected system, configuring the connected system's driver or driver set, viewing the entitlements, and removing the association between the object and the connected system.

Inspecting Objects

You can use the Object Inspector to view detailed information about how an object participates in Identity Manager relationships. These relationships include the connected systems that are associated with the object, how data flows between the Identity Vault and the connected systems, the attribute values that are currently stored in the Identity Vault and on the connected systems, the connected system driver configurations, and so forth.

1. Open the Identity Manager Administration page.
2. In the **Administration** list, click **Object Inspector** to display the Object Inspector page.
3. Specify the fully distinguished name of the object that you want to inspect, or click the browse icon to browse to and select the desired object.

   Identity Console keeps a record of the objects you have previously selected, so you can also use the icon to select from a list of previously selected objects.
4. After you have selected the object, click **OK** to display the Object Inspector page.

   The Connected Systems section lists each of the connected systems with which the object is associated. You can perform any of the following actions:

   - **Delete:** To delete an association with a connected system, select the check box to the left of the association and click **Delete**. To delete all associations, select the check box beneath the Delete column, then click **Delete**.
   - **Refresh:** Select **Refresh** to re-read the connected system associations and refresh the table.
   - **Actions:** Select a connected system by clicking the check box to the left of the association reference (you do not need to select any boxes for the **Add New Association** action item). Click **Actions**, then choose one of the following options:
     - **Run Overview on Driver:** Launches the overview page for the connected system's driver.
     - **Run Overview on Driver Set:** Launches the overview page for the connected system's driver set.
     - **Configure Driver:** Launches the properties page for the connected system's driver so that you can modify the driver's properties.
     - **Configure Driver Set:** Launches the properties page for the connected system's driver set so that you can modify the driver set's properties.

- **Add New Association:** Prompts you for the parameters necessary to add new attribute values to the object's DirXML-Association attribute.
- **Edit Selected Association:** Prompts you to edit the parameters of the connected system's DirXML-Association attribute values.
- **View Entitlements:** Displays a list of the entitlements associated with the connected system. The list displays the current state of the entitlement (granted or revoked) as well as the source of the entitlement (for example, workflow or role-based).

- **Connector:** Lists the connected system's fully distinguished name that is associated with the object. Click the plus (+) icon next to the connected system to see how data flows through the connected system.

  The Servers entry shows the servers that are associated with the driver set. Clicking the **Edit** icon to the right of the server brings up the server's properties page in a pop-up window. Clicking the **Query** icon queries the attribute values for all classes in the driver filter. The larger the filter, the longer the query takes. If the Inspector cannot communicate with the connected system, you see a message stating that the attribute cannot be queried from the application.

  The driver filter's associated classes (such as Group) and their attributes (such as Description and Member) are listed under the Server entry. Click the class to see all of the values for the defined attributes in that class. You can also click an attribute to see its values, or you can click the entries to the right of the attributes to see just the Identity Vault value or the application value. If no value has been defined, the entry displays No Values. If the Inspector cannot communicate with the connected system, you see a message stating that the attribute cannot be queried from the application.
- **States:** The connected system's driver states are Enabled, Disabled, Processed, Pending, Manual, and Migrate.
- **Object ID:** The identification value of the associated object to the connected system. If the connected system driver has no identification, this column displays **None**.

Inspecting Drivers

You can use the Driver Inspector to view detailed information about the objects associated with a driver.

1. Open the Identity Manager Administration page.
2. In the **Administration** list, click **Driver Inspector** to display the Driver Inspector page.
3. In the **Driver to inspect** field, specify the fully distinguished name of the driver that you want to inspect, or click the browse icon to browse to and select the desired driver.

   Identity Console keeps a record of the objects you have previously selected, so you can also use the icon to select from a list of previously selected objects.
4. After you have selected the driver to inspect, click **OK** to display the Driver Inspector page.

   The page displays information about the objects associated with the selected driver. You can perform any of the following actions:

   - **Driver:** Displays the name of the inspected driver. Click the driver name to display the Driver Overview page.
   - **Driver Set:** Displays the name of the driver set in which the inspected driver resides. Click the driver set name to display the Driver Set Overview page.

- **Delete:** Removes the association between the driver and an object. Select the check box in front of the object you no longer want associated with the driver, click **Delete**, then click **OK** to confirm the deletion.
- **Refresh:** Select this option to re-read all of the objects associated with the driver and refresh the information.
- **Show:** Select the number of associations to display per page. You can select a predefined number (25, 50, or 100) or specify another number of your choice. The default is 50 associations per page. If there are more associations than the number displayed, you can use the arrow buttons to display the next and previous pages of associations.
- **Actions:** Perform actions on the objects associated with the driver. Click **Actions**, then select one of the following options:
  - **Show All Associations:** Displays all objects associated with the driver.
  - **Filter for Disabled Associations:** Displays all objects associated with the driver that have a Disabled state.
  - **Filter for Manual Associations:** Displays all objects associated with the driver that have a Manual state.
  - **Filter for Migrate Associations:** Displays all objects associated with the driver that have a Migrate state.
  - **Filter for Pending Associations:** Displays all objects associated with the driver that have a Pending state.
  - **Filter for Processed Associations:** Displays all objects associated with the driver that have a Processed state.
  - **Filter for Undefined Associations:** Displays all objects associated with the driver that have an Undefined state.
  - **Association Summary:** Displays the state of all objects associated with the driver.
- **Object DN:** Displays the DN of the associated objects.
- **State:** Displays the association state of the object.
- **Object ID:** Displays the value of the association.

9 Managing Driver Cache Files

Viewing a Transaction

You can use Identity Console to view the transactions in a driver's cache file. The Driver Cache Inspector displays information about the cache file, including a list of the events to be processed by the driver.

1. Open the Identity Manager Administration page.
2. In the **Administration** list, click **Driver Cache Inspector**.
3. In the **Driver to inspect** field, specify the fully distinguished name of the driver whose cache you want to inspect, or click the browse icon to browse to and select the desired driver, then click **OK** to display the Driver Cache Inspector page.

   A driver's cache file can be read only when the driver is not running. If the driver is stopped, the Driver Cache Inspector page displays the cache as shown in the screen shot below. If the driver is running, the page displays a `Driver not stopped, cache cannot be read` note in place of the cache entries. To stop the driver, click the button; the cache is then read and displayed.

   - **Driver:** Lists the driver that is associated with the cache file. Click the link to display the Driver Overview page.
   - **Driver Set:** Lists the driver set in which the driver resides. Click the link to display the Driver Set Overview page.
   - **Driver's cache on:** Lists the server that contains this instance of the cache file. If the driver is running on multiple servers, you can select another server in the list to view the driver's cache file for that server.
   - **Start/Stop Driver icons:** Displays the current state of the driver and allows you to start or stop the driver. The cache can be read only while the driver is stopped.
   - **Edit icon:** Allows you to edit the properties of the currently selected server.
   - **Delete:** Select entries in the cache, then click **Delete** to remove them from the cache file.
   - **Refresh:** Select this option to re-read the cache file and refresh the information.
   - **Show:** Select the number of entries to display per page. You can select a predefined number (25, 50, or 100) or specify another number of your choice. The default is 50 entries per page. If there are more entries than the number displayed, you can use the arrow buttons to display the next and previous pages.
   - **Actions:** Allows you to perform actions on the entries in the cache file. Click **Actions** to expand the menu, then select one of the following options:
     - **Expand All:** Expands all of the entries displayed in the cache file.
     - **Collapse All:** Collapses all of the entries displayed in the cache file.
     - **Go To:** Allows you to access a specified entry in the cache file. Specify the entry number, then click **OK**.
     - **Cache Summary:** Summarizes all of the events stored in the cache file.

Viewing the Out of Band Sync Cache

To view events in the Out of Band Sync cache:

1. Open the Identity Manager Administration page.
2. In the **Administration** list, click **Driver Out of Band Sync Cache Inspector**.
3. Specify the fully distinguished name of the driver whose cache you want to inspect, or click the browse icon to browse to and select the desired driver, then click **OK**. A driver's cache file can be read only when the driver is not running. For more information, see "Viewing a Transaction" on page 95.

Relocating the Event Cache File

Every driver that is configured in Identity Manager has an associated event cache file. Events are cached in a TAO file before a driver processes them. By default, the TAO files are placed in the `dib` directory.

Identity Manager allows you to place the TAO files anywhere in the file system. Distributing the file I/O across multiple file systems improves the I/O throughput. Each driver can have an optional single-valued server readable attribute `DirXML-CacheLocation`. The value of this attribute is an absolute path to the TAO files in the file system. When the engine is restarted, it looks for the `DirXML-CacheLocation` attribute and the associated TAO files.

You can change the location of the TAO file by using the dxcmd utility:

Run `dxcmd > login > Driver Operations > (Select the Driver) > 14. Cache Operations > 10. Get Cache Path/11. Set Cache Path`

NOTE: You can change the location of the cache file only when the driver is in a disabled state, otherwise it throws an `INVALID_REQUEST` exception. If the path does not exist, it throws a `BAD_FILE_NAME` exception.

To relocate the TAO file:

1. Shut down the driver on which you want to set a new TAO file location.
2. Take note of the system time.

   You might need this data to force resynchronization of objects if any events are missed during the TAO file relocation.
3. Move the driver's TAO file to the new location and disable the driver.
4. Set the new TAO file location by using the `dxcmd` utility.
5. Enable the driver by deselecting the **Do not automatically synchronize** option.
6. Start the driver.
7. Use the **Synchronize** option to force resynchronization of objects from the time noted in Step 2.

10 Securely Storing Driver Passwords with Named Passwords

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

You can add named passwords to a driver set or to individual drivers. Named passwords for a driver set are available to all drivers in the set. Named passwords for an individual driver are available only to that driver.

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 Identity Manager 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.

NOTE: The sample configurations provided for the Identity Manager Driver for Lotus Notes include an example of using named passwords in this way. The Notes driver shim has also been customized to support other ways of using named passwords, and examples of those methods are also included. For more information, see the section on named passwords in the *Identity Manager Driver Guide for Lotus Notes*.

Using Designer to Configure Named Passwords

1. In Designer, select the driver, then right-click and select **Properties**.
2. Select **Named Password**, then click **New**.
3. Specify a name, display name, and a password, then click **OK** twice.

Using Identity Console to Configure Named Passwords

1. Click **IDM Administration** tab from the Identity Console home screen and locate the driver set or driver where you want to add a named password:.
2. To add a named password to a driver set, click the Action menu icon, then click **Driver Set Properties > Named Password** under **Driver Set Configuration**.

   or

   To add a named password to a driver, click the driver object, then click **Configuration** > **Named Passwords**.
3. Click **Add Named Password**.
4. Specify a name, display name and a password, then click **OK** twice.
5. A message appears: `Do you want to restart the driver to put your changes in effect?` Click **OK**.

Using Named Passwords in Driver Policies

Using the Policy Builder

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

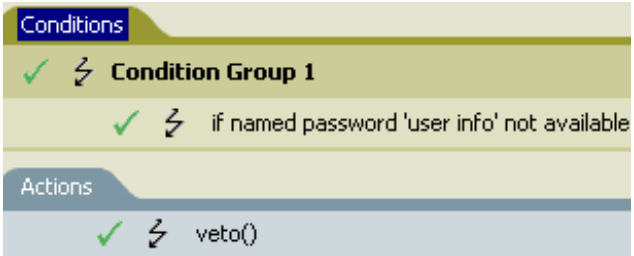
1. In Designer, launch the Policy Builder, right-click, then click **New > Rule**.
2. Specify the name of the rule, then click **Next**.
3. Select the condition structure, then click **Next**.
4. Select **named password** for the **Condition**.
5. Browse to and select the named password that is stored on the driver.

   In this example, it is `user info`.
6. Select whether the operator is available or not available, then click **Next**.
7. Select an action for the **Do** field.

   In this example, the action is `veto`.
8. Click **Finish**.

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

Figure 10-1 *A Policy Using Named Passwords*



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/nxsl/java/
com.novell.nds.dirxml.driver.XdsQueryProcessor"/>
```

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 more information, see Chapter 13, "Using the DirXML Command Line Utility," on page 107.
2. Specify your user name and password.
3. Specify one of the following options:
   - Option 3 for Driver Operations
   - Option 4 for Driver Set Operations

   Option 3 for Driver Operations: If you specified 3, a numbered list of drivers appears. Do the following:

   1. Specify the number for the driver to which you want to add a named password.
   2. Specify 13 for Password Operations.
   3. Specify 5 to set a new named password.

      Go to Step 4.

   Option 4 for Driver Set Operations: If you specified 4, a numbered list of driver set operations appears.

   Do the following:

   1. Specify 5 for Password Operations.
   2. Specify 1 to set a new named password.

      Go to Step 4.
4. At the prompt, specify the name by which you want to refer to the named password.
5. At the prompt, specify a description of the password.
6. At the prompt, specify the actual password that you want to secure.

   The characters you type for the password do not appear on the screen.
7. At the prompt, confirm the password by specifying it again.

   The password operations menu appears.
8. Specify the 99 option twice to exit the menu and quit the DirXML Command Line utility.

Removing a Named Password in the DirXML Command Line Utility

This option is useful if you no longer need named passwords you previously created.

1. Run the DirXML Command Line utility.

   For more information, see Chapter 13, "Using the DirXML Command Line Utility," on page 107.
2. Specify your user name and password.

3 Specify one of the following:

- Option 3 for Driver Operations
- Option 4 for Driver Set Operations

Option 3 for Driver Operations: If you specified 3, a numbered list of drivers appears. Do the following:

1. Enter the number for the driver from which you want to remove named passwords.
2. Specify 13 for Password Operations.
3. (Optional) Specify 7 to see the list of existing named passwords.

   This helps you to make sure that you are removing the correct password.
4. Specify 6 to remove one or more named passwords.
5. Go to Step 4.

Option 4 for Driver Set Operations: If you specified 4, a numbered list of driver set operations appears.

Do the following:

1. Specify 5 for Password Operations.
2. (Optional) Specify 3 to see the list of existing named passwords.

   This helps you to make sure that you are removing the correct password.
3. Specify 2 to remove one or more named passwords.
4. Go to Step 4.

4 At the following prompt, enter No to remove a single named password:

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

5 At the following prompt, enter the name of the named password you want to remove:

```
Enter password name:
```

The password operations menu appears.

6 (Optional) Specify the appropriate number to see the list of existing named passwords.

This step helps you to verify that you have removed the correct password.

7 Specify the 99 option twice to exit the menu and quit the DirXML Command Line utility.

11 Configuring Java Environment Parameters

Rather than use command line options and configuration files to set the environment parameters for the Java virtual machine (JVM) associated with a driver set, you can use Identity Console or Designer.

Using Identity Console to Configure the Java Environment Parameters

1. Click the **IDM Administration** tab from the Identity Console home screen.
2. Click the Action menu icon of the appropriate **Driver Set > Driver Set Properties**.
3. Click **Java Environment Parameters** under **Driver Set Configuration** to display the property page that contains the Java environment parameters.
4. Modify the following settings as desired:

   Classpath Additions: Specify additional paths for the JVM to search for package (`.jar`) and class (`.class`) files. Using this parameter is the same as using the `java -classpath` command. When entering multiple class paths, separate them with a semicolon (;) for a Windows JVM and a colon (:) for a UNIX or Linux JVM.

   JVM Options: Specify additional options to use with the JVM. Refer to your JVM documentation for valid options.

   `DHOST_JVM_OPTIONS` is the corresponding environment variable. It specifies the arguments for JVM 1.2. For example:

   ```
   -Xnoagent -Xdebug -Xrunjdwp: transport=dt_socket,server=y, address=8000
   ```

   Each option string is separated by whitespace. If an option string contains whitespace, then it must be enclosed in double quotes.

   The driver set attribute option has precedence over the `DHOST_JVM_OPTIONS` environment variable. This environment variable is tacked on to the end of driver set attribute option.

   Initial Heap Size: Specify the initial (minimum) heap size available to the JVM. Increasing the initial heap size can improve startup time and throughput performance. Use a numeric value followed by G, M, or K. If no letter size is specified, the size defaults to bytes. Using this parameter is the same as using the `java -Xms` command.

   `DHOST_JVM_INITIAL_HEAP` is the corresponding environment variable. It specifies the initial JVM heap size in decimal number of bytes. It has precedence over the driver set attribute option.

   Refer to your JVM documentation for information about the JVM's default initial heap size.

   Maximum Heap Size: Specify the maximum heap size available to the JVM. Use a numeric value followed by G, M, or K. If no letter size is specified, the size defaults to bytes. Using this parameter is the same as using the `java -Xmx` command.

   `DHOST_JVM_MAX_HEAP` is the corresponding environment variable. It specifies the maximum JVM heap size in decimal number of bytes. It has precedence over the driver set attribute option.

Refer to your JVM documentation for information about the JVM's default maximum heap size.

5 Click **OK** to save your changes.

6 Restart eDirectory to apply the changes.

Using Designer to Configure the Java Environment Parameters

1 Open your project in the Modeler.

2 Right-click the driver set icon , then click **Properties** > **Java**.

3 Modify the following settings as desired:

Server: If the driver set is associated with multiple Identity Manager servers, select the server whose JVM parameters you want to configure.

Classpath Additions: Specify additional paths for the JVM to search for package (`.jar`) and class (`.class`) files. Using this parameter is the same as using the `java -classpath` command. When entering multiple class paths, separate them with a semicolon (;) for a Windows JVM and a colon (:) for a UNIX/Linux JVM.

JVM Options: Specify additional options to use with the JVM. Refer to your JVM documentation for valid options.

`DHOST_JVM_OPTIONS` is the corresponding environment variable. It specifies the arguments for JVM 1.2. For example:

```
-Xnoagent -Xdebug -Xrunjdwp: transport=dt_socket,server=y, address=8000
```

Each option string is separated by whitespace. If an option string contains whitespace, then it must be enclosed in double quotes.

The driver set attribute option has precedence over the `DHOST_JVM_OPTIONS` environment variable. This environment variable is tacked on to the end of driver set attribute option.

Initial Heap Size: Specify the initial (minimum) heap size available to the JVM in bytes. Increasing the initial heap size can improve startup time and throughput performance. Using this parameter is the same as using the `java -Xms` command.

`DHOST_JVM_INITIAL_HEAP` is the corresponding environment variable. It specifies the initial JVM heap size in decimal number of bytes. It has precedence over the driver set attribute option.

Refer to your JVM documentation for information about the JVM's default initial heap size.

Maximum Heap Size: Specify the maximum heap size available to the JVM. Use a numeric value followed by G, M, or K. If no letter size is specified, the size defaults to bytes. Using this parameter is the same as using the `java -Xmx` command.

`DHOST_JVM_MAX_HEAP` is the corresponding environment variable. It specifies the maximum JVM heap size in decimal number of bytes. It has precedence over the driver set attribute option.

Refer to your JVM documentation for information about the JVM's default maximum heap size.

4 Click **OK** to save your changes.

5 To deploy the changes into your Identity Vault, right-click the driver set icon , click **Live** > **Deploy**, and follow the deployment prompts.

6 Restart eDirectory to apply the changes.

12 Rights Needed by a Driver on Identity Vault Objects

An Identity Manager driver requires the following minimum rights to the Identity Vault objects:

- Read rights to the attributes in the Subscriber channel filter. The driver needs these rights at least to the objects within the scope of objects that the driver will be working on.
- Read and Write rights to all objects and attributes in the Publisher filter for the scope of the objects that the driver works with. The driver must have Read rights to the objects outside its scope for appropriate matching rules.
- Read rights to passwords of objects in the driver scope to set passwords.
- Read rights to the DirXML Script policy objects.
- Read and Write rights to the driver objects and objects under it for updating the following attributes:
  - DirXML-DriverStorage attribute on the driver object (resides in the driver object and regularly modified)
  - DirXML-State attribute (modified by the driver operation)
  - DirXML-StatusLog attribute (resides in driver, Publisher channel, and Subscriber channel objects)

13 Using the DirXML Command Line Utility

The DirXML Command Line utility allows you to use a command line interface to manage the driver. The primary use of this utility is to allow you to create platform-specific scripts to manage the driver.

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`
- UNIX/Linux: `/opt/novell/eDirectory/bin/dxcmd`

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

- Interactive mode
- Command line mode

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 13-1 on page 107 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 13-1 *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.

Option	Description
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. For a list of operations, see Table 13-2 on page 108.
4: Driver set operations	Lists the operations available for the driver set. For a list of operations, see Table 13-3 on page 111.
5: Log events operations	Lists the operations available for logging events through Audit. For a description of these options, see Table 13-6 on page 114.
6: Get DirXML version	Lists the version of Identity Manager installed.
7: Job operations	Manages jobs created for Identity Manager.
8: JVM Statistics	Lists the performance statistics such as, memory, thread, runtime, classloader, garbage collection and OS information for an instrumented Java Virtual Machine (JVM).
99: Quit	Exits the DirXML Command Line utility.

Table 13-2 *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

Options	Description
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 NetIQ `nds.dtd` (https://www.netiq.com/documentation/identity-manager-developer/dtd-documentation/ndsdtd/query.html). Examples: NetWare: `sys:\files\query.xml` Windows: `c:\files\query.xml` Linux: `/files/query.xml`

8: **Submit XDS command document to driver**	Submits an XDS command document to the driver's Subscriber channel, bypassing the driver cache. The document is processed before 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.
	
	`Enter filename of XDS command document:`
	
	Examples:
	
	Windows: `c:\files\user.xml`
	
	Linux: `/files/user.xml`
	
	`Enter name of file for response:`
	
	Examples:
	
	Windows: `c:\files\user.log`
	
	Linux: `/files/user.log`

Options	Description
9: Submit XDS event document to driver	Submits an XDS event document to the driver's Subscriber channel, bypassing the driver cache. The document is processed before 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. `Enter filename of XDS event document:` Examples: Windows: `c:\files\add.xml` Linux: `/files/add.xml`
10: Queue event for driver	Submits a document to the driver's Subscriber channel by queuing the document in the driver cache. The document is processed after anything that might be in the cache at the time of the submission. The submission does not fail if the driver is not running. `Enter filename of XDS event document:` Examples: 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 13-4 on page 112 for a description of these options.
14: **Cache operations**	There are five Cache operations. See Table 13-5 on page 113 for a descriptions of these options.
99: **Exit**	Exits the driver options.

Sample XDS Event Document

```
<nds dtdversion="1.1" ndsversion="8.6" xml:space="default">
  <input>
    <add class-name="User" src-dn="Doe John">
      <association>JDoe@novell.com</association>
      <add-attr attr-name="LastName">
        <value type="string">John</value>
      </add-attr>
      <add-attr attr-name="FirstName">
        <value type="string">Doe</value>
      </add-attr>
      <add-attr attr-name="Email">
        <value type="string">JDoe@novell.com</value>
      </add-attr>
    </add>
  </input>
</nds>
```

Sample XDS Command Document

```
<nds dtdversion="3.5" ndsversion="8.x">
  <source>
    <product version="3.5.11.4223">DirXML</product>
    <contact>Novell, Inc.</contact>
  </source>
  <input>
    <add cached-time="20080519102858.809Z" class-name="User" eventid=
      "blr-krajiv-sles#20080519102858#1#1" qualified-srcdn=
      "O=n\OU=People\CN=JDoe" src-dn="\KRAJIV-LINUXTREE\n\People\JDoe"
      src-entry-id="32956" timestamp="1211192938#9">
      <add-attr attr-name="Internet EMail Address">
        <value timestamp="1211192938#8"
          type="string">JDoe@novell.com</value>
      </add-attr>
      <add-attr attr-name="Given Name">
        <value timestamp="1211192938#5" type="string">John</value>
      </add-attr>
      <add-attr attr-name="Surname">
        <value timestamp="1211192938#9" type="string">Doe</value>
      </add-attr>
    </add>
  </input>
</nds>
```

Table 13-3 *Driver Set Operations*

Operation	Description
1: Associate driver set with server	Adds a driver set to the server after which the driver set becomes active.
2: Disassociate driver set from server	Removes a driver set from the server after which the driver set becomes inactive.

Operation	Description
3: Export Identity Manager server public key certificate	Exports the DirXML server's public key certificate which is used for encrypting data when setting passwords.
4: Regenerate Identity Manager server keypair	Makes the DirXML Engine regenerate the public key/ private key pair which is used for encrypting data when setting passwords.
5: Passwords operations	There are four password operations. For description of these operations, see the operations 5, 6, 7, and 99 in the Table 13-4 on page 112.
6: Get default reciprocal attribute mappings	Lists the default reciprocal attribute mappings.
7: Regenerate all Identity Manager server keys	Makes the DirXML Engine regenerate all server-specific encryption keys.
8: Apply Activation	Activates the Identity Manager Engine and Drivers depending on the activation file you select.
9: View Activation	Displays the existing activation information for Identity Manager Engine and drivers.
99: > Exit	Exits the current menu and takes you back to the DirXML commands.

Table 13-4 *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.

Operation	Description
5: **Set named password**	Allows you to store a password or other pieces of security information about the driver. For more information, see Chapter 10, "Securely Storing Driver Passwords with Named Passwords," on page 97. 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.
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.

Table 13-5 *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.

Operation	Description
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 position token** (default=0): ◆ **Enter maximum transactions records to return** (default=1): ◆ **Enter name of file for response:**
4: **Delete cached transactions**	Deletes the transactions stored in the 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.

NOTE: In the same dxcmd session, if you wish to view the cached transactions after deleting few transactions, you need to reset the position value to 0 rather than accepting the default value. If you accept the default value, you may receive an `ERR_INVALID_REQUEST` exception.

Table 13-6 *Log Events Operations*

Operation	Description
1: **Set driver set log events**	Allows you to log driver set events through Audit. There are 49 items you can select to log. See Table 13-7 on page 115 for a list of these options. Enter 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 Audit. There are 49 items to select to log. See Table 13-7 on page 115 for a list of these options. Enter 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 13-7 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 13-8 Job Operations

Options	Description

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

2: **Operations on specific job object**	Allows you to perform operations for a specific job. `Enter the job number:` The following list of options appears: **1: Send job update notification** **2: Start job** **3: Abort running job** **4: Get job state** **5: Check job configuration** **6: Passwords operations** **99: Exit**

Command Line Mode

The command line mode allows you to use script or batch files. Table 13-9 on page 117 contains 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 novell -start test.driverset.headquarters`

This example command starts the driver.

Table 13-9 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.

Option	Description
-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.
-cert *<X.509 DER certificate filename>*	Certificate file used for encrypting passwords.
-version *<n.n[.n[.n]]>*	Changes engine version by force.
-nossl	Uses clear socket for LDAP.
-keystore *<keystore path and filename>*	Specifies the filename of the Java keystore that contains the trusted root certificate of the issuer of the certificate used by the remote interface shim.
-storepass *<keystore password>*	Specifies the password for the Java keystore specified by the keystore parameter.
-dnform *<slash|qualified-slash|dot|qualified-dot|ldap>*	Changes the dn form by force.
Actions	
-start *<driver dn>*	Starts the driver.
-stop *<driver dn>*	Stops the driver.
-getstate *<driver dn>*	Returns the value that indicates the state of the driver (0 - stopped, 2 - running, and so on).
-getdriverstats *<driver dn> <output filename>*	Shows the statistics of the driver.
-resetdriverstats *<driver dn>*	Resets the statistics of the driver.
-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.
-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.

Option	Description
-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 NetIQ `nds.dtd` (https://www.netiq.com/documentation/identity-manager-developer/dtd-documentation/ndsdtd/).
-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>*	Submits a document to the driver's Subscriber channel, bypassing the driver cache. The document gets 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. 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 gets 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.
-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.

Option	Description
-setlogevents *<dn> <integer ...>*	Sets Audit log events on the driver. The integer is the option of the item to log. See Table 13-7 on page 115 for the list of the integers to enter.
-clearlogevents *<dn>*	Clears all 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.
-getjvmstats *<driver dn><output file>*	Shows details of memory, thread, runtime, classloader, garbage collection and OS that is installed.
-updatejob *<job dn>*	Updates the specified job.
-clearallnamedpaswords *<driver dn>*	Clears all named passwords set on a specific driver.
-applyactivation *<driverset dn> <activation file>*	Activates the specified driver or Identity Manager engine
-viewactivation *<driverset dn> <output filename>*	Displays the activation information for the specified driver or Identity Manager engine
-exportcerts *<kmoname> <server|client> <java|native|dotnet> <output dir>*	Exports the certificates from eDirectory KMO in different formats based on the type of driver shim loaded by Remote Loader.

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 13-10 on page 121 contains other values for specific command line options.

Table 13-10 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. The getstate option returns the state of the driver. It does not show the state. You can access the return value by using '$?' in UNIX/ Linux and '%errorlevel%' in Windows. The return value can be used in a batch or shell script.
-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	The return is the next scheduled time for the job in eDirectory time format (number of seconds since 00:00:00 Jan 1, 1970 UTC).

14 Synchronizing Objects

The following sections explain how data is synchronized between the Identity Vault and connected systems.

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 all Identity Vault objects and classes that are included in the filter on the Subscriber channel.
- 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.

When Is Synchronization Done?

The Identity Manager engine performs object synchronization or merging in the following 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 Identity Manager engine submits a <sync> event element for each object found as the result of a migrate-into-NDS query. These <sync> events are submitted by 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 Identity Manager 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 Identity Console driver's overview page.
- The Identity Manager 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 "How Does the Identity Manager Engine Decide Which Object to Synchronize?" on page 124.

How Does the Identity Manager Engine Decide Which Object to Synchronize?

The Identity Manager 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 in 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 cached events that have not 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 Identity Manager 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.
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.

How Does Synchronization Work?

After the Identity Manager 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.x and Identity Manager 3.x.
   - The connected system is queried for all values in the Publisher filter, and for values that are marked for synchronization in Identity Manager 2.x and Identity Manager 3.x.
2. The returned attribute values are compared and modification lists are prepared for the Identity Vault and the connected system according to Table 14-1 on page 126, Table 14-2 on page 127, and Table 14-3 on page 128.

   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, it 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.

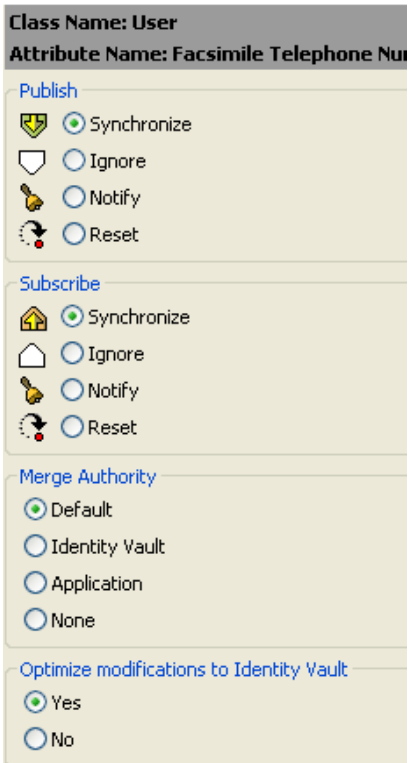
There are three different combinations of selected items in the filter, and each one creates a different output.

- "Scenario One" on page 125
- "Scenario Two" on page 127
- "Scenario Three" on page 128

Scenario One

The attribute is set to **Synchronize** on the Publisher and Subscriber channels, and the merge authority is set to **Default**.

Figure 14-1 Scenario One



The following table contains the values that the Identity Manager 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.

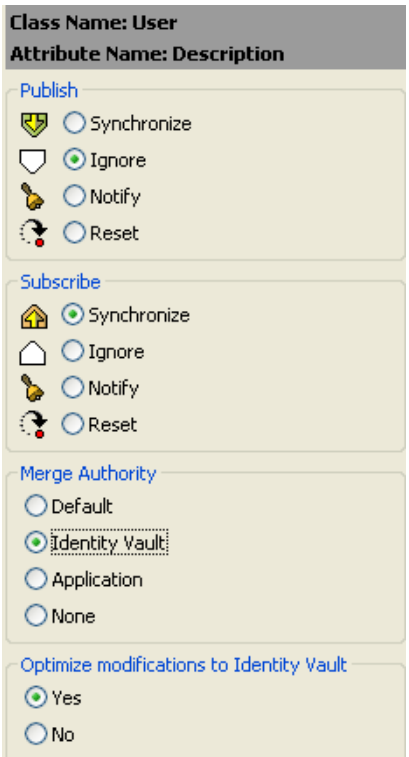
Table 14-1 Output of Scenario One

	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

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. The merge authority is set to **Identity Vault**.

Figure 14-2 *Scenario Two*



The following table contains the values that the Identity Manager 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.

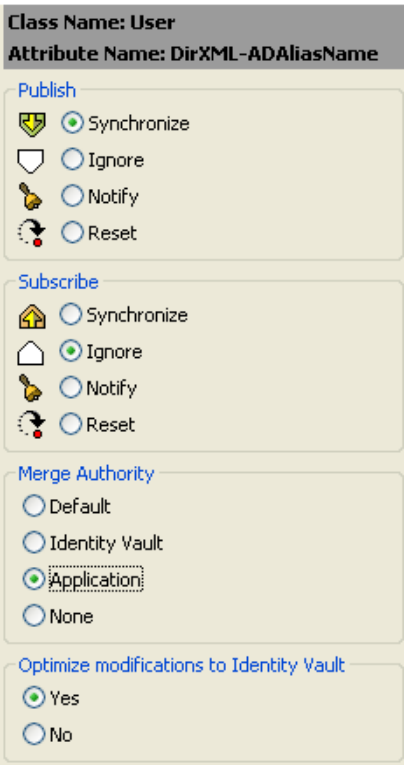
Table 14-2 *Output of Scenario Two*

	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 non-empty	App = empty	App = Identity Vault	Identity Vault = App	App = Identity Vault
Application multi-valued empty	No change	App = Identity Vault	No change	App = Identity Vault
Application multi-valued non-empty	App = empty	App = Identity Vault	App = empty	App = Identity Vault

Scenario Three

The attribute is set to **Synchronize** on the Publisher channel or the merge authority is set to **Application**.

Figure 14-3 *Scenario Three*



The following table contains the values that the Identity Manager 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 14-3 *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

15 Association Statistics

By using the Identity Manager Association Statistics feature, you can find the association details of the identities managed by Identity Manager. Identity Manager uses the association statistics to obtain the association count for the Identity Manager drivers.

To obtain active, inactive, and system managed objects for a driver, run the association statistics job. You can schedule the association statistics job on a daily, weekly, monthly, or yearly basis. By default, the job is scheduled to run every week. To modify the configuration setting for this job, use Identity Console. For more information, see "Modifying the Association Statistics Job Configuration" in the *NetIQ Identity Manager Jobs Guide*.

The Association Statistics dashboard displays the association details. Alternatively, you can view the details by exporting the associations to a file.

NOTE:

- The association count for the drivers is per server. If an object is associated with more than one driver, the association count is calculated uniquely for each driver.
- If you have more than 200,000 associations, NetIQ recommends you to set the maximum heap size for the driverset to 2 GB or more. For information about setting the heap size, see "Using Identity Console to Configure the Java Environment Parameters" on page 101.

To view the association statistics:

1. In Identity Console, select **IDM Administration > Click on the Action menu of the appropriate Driver Set > Driver Set Properties > Inspector and Statistics > Association Statistics**.
2. Select the server for which you want to run the association statistics.
3. The association count displays the previously computed result.

   Identity Console displays the association count for active, inactive, and system managed objects for all the drivers associated with the driverset.

   Identity Console considers groups and organization units as system managed objects. Identity Console considers an object inactive, if the `Login Disabled` attribute in the object is set to true and the object has not been modified within the last 120 days. All the remaining objects are considered as active managed objects.
4. Click **Recompute** icon to obtain the updated results.

   When a driver is disabled on the server, Identity Console does not display the driver in the dashboard.
5. Click **Export Statistics** icon to export the system details and association count details for the drivers associated with the server.
6. To export the objects associated with a specific driver, click the **Export Statistics** icon next to the required objects and save the file.

NOTE: In case of Fan-Out drivers, only unique objects are exported. If an object is associated with multiple instances of a Fan-Out driver, Identity Console displays all the association counts in the dashboard. However, if you choose to export the objects in a file, Identity Console exports only the unique objects.

7 Click **Actions** and select the required option to organize the association count dashboard.

16 Enabling Out of Band Sync

The Identity Manager drivers process events in the order they occur, which guarantees that all changes required for an event to successfully process are applied in the order they occur.

However, there are instances when you want a certain event to take precedence over others. For example, events that involve password changes, locking an account, or disabling an account should take precedence over other events. Identity Manager provides the Out of Band Sync feature that allows you to assign a higher priority to these events, so that they are processed before other events in the queue.

When you enable this feature for an attribute, Identity Manager creates a new cache for the driver for storing the Out of Band Sync events. You can enable or disable this functionality, by using the new setting included in the driver filter. Also, Identity Manager creates an Out of Band Sync status cache for maintaining the status of events, which are synchronized from the Out of Band Sync sync cache. By maintaining the status of events in the Out of Band Sync status cache, Identity Manager drivers avoid duplication of events or sending of old events from the base driver cache.

Enabling Out of Band Sync Using Designer

To enable this feature, open a driver's filter, select the desired attribute, and then set the option **Perform Out of Band Sync (Subscriber)** to Yes. This option is available only for attributes, not classes, and is set to No by default.

Specifying the Out of Band Sync Status Interval

When you enable Out of Band Sync, the driver maintains a status cache to store the status of the events that are successfully processed from the Out of Band cache. This status cache is used to ensure that duplicate or old events are not sent across when events from the normal driver cache are processed.

The Out of Band Sync status cleanup interval specifies the time in minutes, after which the entries in the event out of band sync status cache are checked for cleanup. This cleans up only the status entries of those events that are already processed from the normal cache. It takes effect only if you enable Out of Band Sync.

The driver includes a new GCV for the Out of Band Sync status cleanup interval on the DriverSet, namely `dirxml.engn.ps.stat.purge.interval`. This GCV can take a minimum value of 1 and a maximum value of 300 minutes. The default value is 5 minutes. If this GCV does not exist on the Driverset, the driver assumes a default value of 5 minutes.

You can use the `dirxml.engn.ps.stat.purge.interval` GCV to set the Out of Band Sync status cleanup interval, as shown in this example:

```
<definition critical-change="true" display-name="Out of Band Sync status
purge
interval" name="dirxml.engn.ps.stat.purge.interval" range-hi="300" range-
lo="1"
type="integer">
<description>Specify the time in minutes, after which the entries in the
Out of Band Sync status cache will be checked for cleanup.</description>
<value>5</value>
</definition>
```

For information about how to view the Out of Band Sync cache, see "Viewing the Out of Band Sync Cache" on page 96.

17 Migrating and Resynchronizing Data

Identity Manager synchronizes data when the data changes. If you want to synchronize all data immediately, you can choose from the following options in Identity Console:

- **Migrate Data from Identity Vault:** Allows you to select containers or objects you want to migrate from the Identity Vault to an application. When you migrate an object, the Identity Manager engine applies all of the Matching, Placement, and Create policies, as well as the Subscriber filter, to the object.
- **Migrate Data into Identity Vault:** Assumes that the remote application can be queried for entries that match the criteria in the Publisher filter.
- **Synchronize:** The Identity Manager engine looks in the Subscriber class filter and processes all objects for those classes. Associated objects are merged. Unassociated objects are processed as Add events.

To use one of the options described above:

1. In the **Roles and Tasks** view, click **Identity Manager > Identity Manager Overview**.
2. Browse to and select the driver set where the driver exists, then click **Search**.
3. Click the driver icon, then click the **Migrate** tab.
4. Click the appropriate migration button.

For more information, see Chapter 14, “Synchronizing Objects,” on page 123.

18 Configuring Stronger Ciphers for SSL Communication

You can configure Identity Manager in Suite B mode to enhance the security requirements of your Identity Manager environment.

Suite B requirement originated from the National Security Agency (NSA) to specify a cryptographic interoperability strategy. Suite B includes the following cryptographic algorithms:

- Encryption based on the Advanced Encryption Standard (AES) using 128-bit keys or 256-bit keys
- Digital signatures with the Elliptic Curve Digital Signature Algorithm (ECDSA) on P-256 and P-384 curves
- Key exchange, either pre-shared or dynamic, using the Elliptic Curve Diffie-Hellman (ECDH) method on P-256 and P-384 curves
- Hashing (digital fingerprinting) based on the Secure Hash Algorithm-2 (SHA-256 and SHA-384)

NOTE: Suite B standard is subject to change. NSA may change their recommendations in future. Suite B support in Identity Manager is based on our interpretation of the NSA recommendations. For more information about Suite B, see Suite B Cryptography.

Prerequisites

To configure Identity Manager in Suite B mode, your environment must meet the following conditions:

- eDirectory 9.0.2 or later is installed as an Identity Vault
- TLS 1.2 is enforced as a communication protocol
- Suite B connection parameter is specified in the driver, Remote Loader, or Fan-Out configuration to enforce the Suite B specification for a secured communication

  NOTE: In Suite B mode, the SSL connection is restricted to accept only Suite B supported certificates. If a certificate is expired or invalid, the handshake fails and the communication is not established. For generating Suite B certificates, see “Creating a Server Certificate Object” in the *NetIQ eDirectory Administration Guide*.

The following table lists the requirements as specified by Suite B:

Requirement	Description
Protocol	TLS 1.2 is supported in Suite B mode.
Public keys	The public key for certificates must be a minimum size of EC 256 bits.

Requirement	Description
Signature algorithm	The signature algorithm for certificates must be a minimum size of ECDSA 256 bits (curve P256) and SHA256.
Hash algorithm	The hash algorithm must have the minimum size of SHA256.
Cipher specification	The following ciphers are supported for Suite B mode: ◆ TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 ◆ TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 To use ciphers with stronger signature and hash algorithms, the certificates of server key file must contain similar or stronger signature and hash algorithms. Suite B supports two levels of cryptographic security: 128 bit and 192 bit. The level defines a minimum strength that all cryptographic algorithms must provide. In Suite B 192-bit processing mode, the supported cipher suite is TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384.

Configuring the Settings for Suite B Mode

To meet the requirements specified by Suite B, you must specify the appropriate settings in the Identity Manager components. This section provides information about those settings.

Engine

To configure the Identity Manager engine in Suite B mode, you must set the Suite B configuration option **enforceSuiteB** = **true** in the driver configuration by using Designer or Identity Console.

With the use of stronger ciphers in Suite B mode, passwords managed by the engine such as named password, application password, and Remote Loader password will be re-encrypted when they are used for the first time after upgrading the engine to 4.6 version. On an upgraded engine, the existing encrypted attribute values in the driver cache file are not re-encrypted with stronger ciphers because they are removed from the TAO file when the event is processed. However, when new encrypted attributes are stored in the cache, they are encrypted with AES 256-bit keys.

Engine and Remote Loader Communication

To make the engine and Remote Loader communication compliant with Suite B mode, set the Suite B configuration option **enforceSuiteB** = **true** in the driver configuration. The Suite B communication can also be configured in the Remote Loader configuration file for a driver by setting **enforceSuiteB** to `true`. For more information, see "Configuring the Remote Loader and Drivers" on page 30.

Suite B mode is disabled by default. When you enable it, Identity Manager automatically uses TLS 1.2 or later for communication. If you try to connect a Suite B-enabled engine with a Remote Loader that does not support TLSv1.2, the handshake fails and the communication is not established. For example, Remote Loader 4.5.3, which does not support TLS v1.2.

Engine and Fan-Out Agent Communication

For enabling the Suite B communication, manually include `netiq.fanoutagent.connection.enforceSuiteB=true` parameter in the Fan-Out Agent configuration file. You also need to specify **enforceSuiteB = true** in the driver configuration. Suite B configuration is supported with driver version 1.0.1.1. For more information, see the *NetIQ Identity Manager Driver for JDBC Fanout Implementation Guide (https://www.netiq.com/documentation/identity-manager-49-drivers/jdbc_fanout/data/netiq-identity-manager-for-jdbc-fan-out-driver-implementation-guide.html)*.

Identity Manager Drivers

Along with the configuration changes discussed in the earlier sections, additional changes have been made to these drivers to enable them for Suite B.

eDirectory to eDirectory Driver

The eDir-to-eDir Driver Certificates Wizard in Designer allows the use of stronger ciphers for encrypting the data as specified by Suite B. You import the Suite B compliant certificates into the certificate store that the driver uses. For more information, see Securing Driver Communication (https://www.netiq.com/documentation/identity-manager-49-drivers/edirectory/data/securing-driver-communication.html) in the *NetIQ Driver for eDirectory Implementation Guide (https://www.netiq.com/documentation/identity-manager-49-drivers/edirectory/data/bookinfo.html)*.

Active Directory Driver

The driver stores the password in the Windows registry. For Suite B compliance, the driver uses AES 256-bit encryption algorithm to encrypt the new passwords.

Passwords that are already in the registry are not re-encrypted with stronger ciphers because they are cleaned up when the event is processed. However, when new passwords are stored in the registry, they are encrypted with AES 256-bit keys.

Enabling Stronger Ciphers for SSL Communication

By default, Identity Manager supports the 128-bit SSL communication between the engine and the Remote Loader/ Fan-Out agent. The supported ciphers include:

- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

Oracle provides a default cryptographic jurisdiction policy file that limits the strength of cryptographic algorithms. When using stronger ciphers, you must increase the strength of encryption used. Cipher suites using key lengths greater than 128 bits, such as 256-bit AES encryption, require the JCE Unlimited Strength Jurisdiction policy files that enable additional cipher suites for Java in a separate JAR file.

To enable 256-bit or higher ciphers:

1. Download and extract the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files zipped file from Oracle's Java website to a temporary folder on your computer.

   For example, download Java 8 JCE files from Oracle’s download page.

2. Navigate to the JRE path of your Identity Manager installation directory and save the `local.policy.jar` and `US_export_policy.jar` files to a different directory.

   For example: `/opt/novell/eDirectory/lib64/nds-modules/jre/lib/security`

3. Replace these policy jars with the files you extracted in Step 1.

   For detailed instructions, see the steps listed in the `Readme.txt` file included in the zipped file.

Verifying the Suite B Settings

When Suite B mode is enabled, the trace shows the cipher values and the TLS version that is used in the SSL communication.

To verify whether the communication is successful in Suite B mode, include a non-EC certificate in the configuration and verify that the trace file messages indicate that the Suite B certificate is not imported. For example, when Suite B is successfully enabled, the trace file can show entries similar to this:

```
[11/28/16 16:34:14.828]:Delimited Text PT:JSSEKMO: [KMO2Keystore:readKMOFromIDV] - Created jclient context : Context=8847472, epoch=1
[11/28/16 16:34:14.829]:Delimited Text PT:JSSEKMO: [KMO2Keystore:readKMOFromIDV] - Id resolution successful. Entry Id : 34269
[11/28/16 16:34:14.829]:Delimited Text PT:JSSEKMO: [KMO2Keystore:readKMOFromIDV] - Successfully read the KMO attributes from IDV
[11/28/16 16:34:14.829]:Delimited Text PT:JSSEKMO: [KMO2Keystore:readKMOFromIDV] - Initialized public key bytes..
[11/28/16 16:34:14.829]:Delimited Text PT:JSSEKMO: [KMO2Keystore:readKMOFromIDV] - Initialized private key bytes..
[11/28/16 16:34:14.829]:Delimited Text PT:JSSEKMO: [KMO2Keystore:readKMOFromIDV] - Freeing up the context.
[11/28/16 16:34:14.830]:Delimited Text PT:JSSEKMO: [KMO2Keystore:parseCertificate] - Certificate parsed successfully...
[11/28/16 16:34:14.830]:Delimited Text PT:JSSEKMO: [KMO2Keystore:initPrivateKey] - Successfully unwrapped the secret.
[11/28/16 16:34:14.830]:Delimited Text PT:JSSEKMO: [KMO2Keystore:initPrivateKey] - cleaning up context.
[11/28/16 16:34:14.831]:Delimited Text PT:JSSEKMO: [KMO2Keystore:setupKeystore] - Initialized the keystore.
[11/28/16 16:34:14.831]:Delimited Text PT:JSSEKMO: [KMO2Keystore:setupKeystore] - keystore load completed.
[11/28/16 16:34:14.831]:Delimited Text PT:JSSEKMO: [KMO2Keystore:setupKeystore] - Instantiated certificate factory.
[11/28/16 16:34:14.832]:Delimited Text PT:JSSEKMO: [KMO2Keystore:setupKeystore] - Successfully setup the certificate chain.
[11/28/16 16:34:14.832]:Delimited Text PT:JSSEKMO: [KMO2Keystore:initKsWithKeyPair] - Successfully generated private key from bytes.
[11/28/16 16:34:14.839]:Delimited Text PT:JSSEKMO: [KMO2Keystore:initKsWithKeyPair] - Successfully created a keystore entry.
[11/28/16 16:34:14.840]:Delimited Text PT:Remote Interface Driver: Creating an JSSEKmoFactory ServerSocket
[11/28/16 16:34:15.109]:Delimited Text PT:Remote Interface Driver: Cipher Suite : TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
[11/28/16 16:34:15.110]:Delimited Text PT:Remote Interface Driver: JSSE Socket, cipher suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 , peer host:
net.ssl.SSLPeerUnverifiedException: peer not authenticated
[11/28/16 16:34:15.110]:Delimited Text PT:Remote Interface Driver: Connection established...
```

In case of an error, you will see messages similar to this:

```
DirXML Log Event -------------------
     Driver:   \SLES17885\system\Driver Set\Delimited Text Driver
     Channel:  Publisher
     Status:   Error
     Message:  java.io.IOException: Suite B certificate is not imported
        at com.novell.nds.dirxml.remote.SocketStream.connect(SocketStream.java:629)
        at com.novell.nds.dirxml.remote.Connection.connectStream(Connection.java:710)
        at com.novell.nds.dirxml.remote.Connection.connect(Connection.java:386)
        at com.novell.nds.dirxml.remote.driver.PublicationShimImpl.start(PublicationShimImpl.java:113)
        at com.novell.nds.dirxml.engine.Publisher.run(Publisher.java:607)
        at java.lang.Thread.run(Thread.java:745)
```

If the certificate is not valid, the trace shows messages similar to this:

```
[02/23/17 19:46:52.294]:Delimited Text Driver PT:JSSEKMO: [KMO2Keystore:parseCertificate] - Certificate parsed successfully...
[02/23/17 19:46:52.302]:Delimited Text Driver PT:JSSEKMO: [KMO2Keystore:initPrivateKey] - Successfully unwrapped the secret.
[02/23/17 19:46:52.303]:Delimited Text Driver PT:JSSEKMO: [KMO2Keystore:initPrivateKey] - cleaning up context.
[02/23/17 19:46:52.303]:Delimited Text Driver PT:JSSEKMO: [KMO2Keystore:setupKeystore] - Initialized the keystore.
[02/23/17 19:46:52.310]:Delimited Text Driver PT:JSSEKMO: [KMO2Keystore:setupKeystore] - keystore load completed.
[02/23/17 19:46:52.314]:Delimited Text Driver PT:JSSEKMO: [KMO2Keystore:setupKeystore] - Instantiated certificate factory.
[02/23/17 19:46:52.316]:Delimited Text Driver PT:JSSEKMO: [KMO2Keystore:setupKeystore] - Successfully setup the certificate chain.
[02/23/17 19:46:52.317]:Delimited Text Driver PT:JSSEKMO: [KMO2Keystore:initKsWithKeyPair] - Successfully generated private key from bytes.
[02/23/17 19:46:52.322]:Delimited Text Driver PT:JSSEKMO: [KMO2Keystore:initKsWithKeyPair] - Successfully created a keystore entry.
[02/23/17 19:46:52.333]:Delimited Text Driver PT:Remote Interface Driver: Creating an JSSEKmoFactory ServerSocket
[02/23/17 19:46:52.393]:Delimited Text Driver PT:Receiving DOM document from application.
[02/23/17 19:46:52.394]:Delimited Text Driver PT:
<nds dtdversion="4.0" ndsversion="8.x">
  <input>
    <status level="error" type="remoteloader">java.io.IOException: Error during SSL handshake
        at com.novell.nds.dirxml.remote.SocketStream.connect(SocketStream.java:619)
        at com.novell.nds.dirxml.remote.Connection.connectStream(Connection.java:710)
        at com.novell.nds.dirxml.remote.Connection.connect(Connection.java:386)
        at com.novell.nds.dirxml.remote.driver.PublicationShimImpl.start(PublicationShimImpl.java:113)
        at com.novell.nds.dirxml.engine.Publisher.run(Publisher.java:607)
        at java.lang.Thread.run(Thread.java:745)
</status>
  </input>
</nds>
```

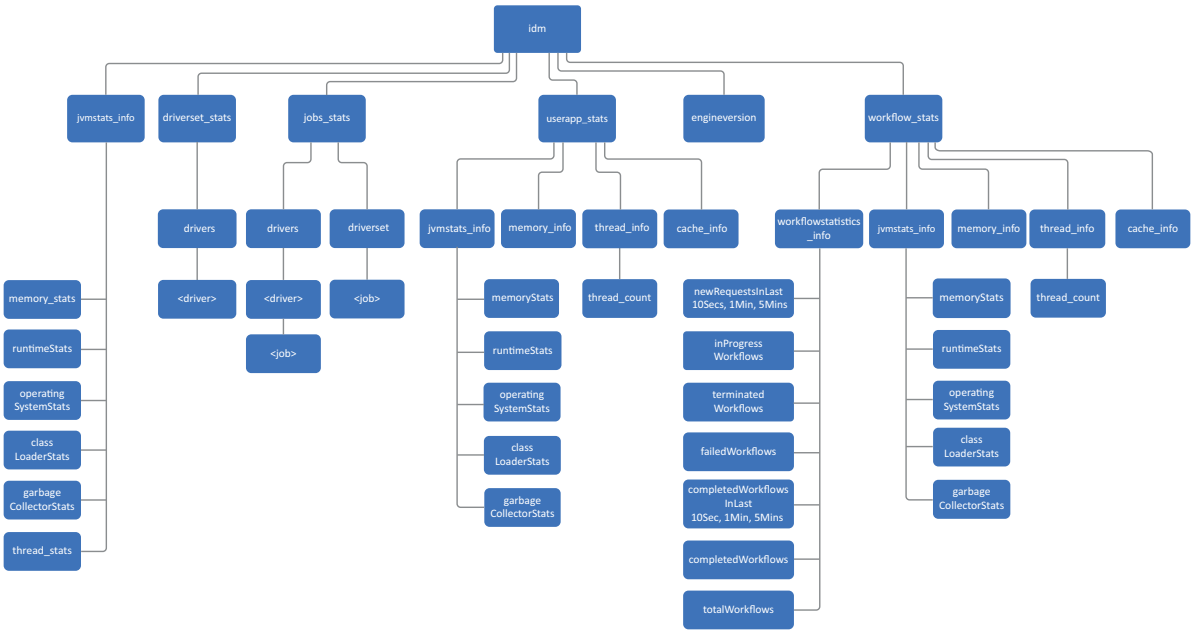
19 Monitoring Identity Manager

Identity Manager leverages eDirectory monitoring framework and provides an LDAP search method for monitoring the state of Identity Manager engine and the health of User Application in your environment. Identity Manager is registered as a data producer in the monitoring framework. Monitoring is supported with eDirectory 9.0.2 and later.

You can obtain the monitoring data by issuing a search request with a search base of `cn=idm,cn=monitor` on the Identity Vault. Using this method provides several advantages. The search is quick and can be embedded in a script for gathering monitoring data at regular intervals. Also, you can consolidate the monitoring data into one common place and format.

IMPORTANT: `cn=idm,cn=monitor` is a virtual object and standardized on the OpenLDAP implementation. This object does not actually reside in the Identity Vault. You use this method for monitoring Identity Manager through LDAP interfaces only.

A subset of the virtual `cn=idm,cn=monitor` objects is shown below.



You can use this hierarchy to construct a searchbase. For example, to monitor the statistics of a driver, start the search from the driver up to the root node. The searchbase will look like this: `cn=<CN of the driver>,cn=drivers,cn=driverset_stats,cn=idm,cn=monitor`

When a search is issued, the monitoring framework generates and returns dynamic objects in LDAP object format. The search response is structured to create a hierarchy of objects, where `cn=idm,cn=monitor` is the root object. For information about the Identity Manager components that can be monitored, see "Viewing the Monitoring Statistics" on page 142.

You can use LDAP clients to access information provided by the monitoring framework, subject to access and other controls, such as LDAP server specific information or connection-specific information. Identity Manager restricts this search only to users with write rights to the `NDSRightsToMonitor` attribute on the NCP server object in eDirectory. This attribute is not populated by default. Therefore, only an administrator or a supervisor of the NCP server can run the search. For information about changing the rights, see the *eDirectory Administration Guide*.

Viewing the Monitoring Statistics

The following table provides information about the components and the statistics that can be monitored:

Components Monitored	Description
jvm_stats	Provides the following information for Java Virtual Machine (JVM) of the system running Identity Manager. ◆ memory_stats: Provides information about the usage of heap and non-heap memory. ◆ thread_stats: Provides information for all live threads, such as name, state, and the count of active threads. ◆ runtime_stats: Provides information about the JVM environment, such as start time, uptime, and system properties. ◆ operating_system_stats: Provides information about the underlying operating system on which Identity Manager is installed, such as name, version, processor, and architecture. ◆ classloader_stats: Provides information about the class loader of the JVM, such as number of loaded classes, unloaded classes, and total number of classes. ◆ garbage_collection_stats: Provides information about the total number of collections that have occurred and the total collection time in milliseconds.
job_stats	Provides the following information: ◆ driverset: Provides information about a specific job or all jobs configured for a driver set. ◆ drivers: Provides information about all jobs under all drivers, all jobs under a specific driver, or a specific job under a specific driver.
driverset_stats	Provides information about the driver set such as driver set DN, the number of drivers configured, driver state count, and startup option count. For each driver, information such as driver DN, state, startup option, processed operation count is provided.
engine version	Version of the Identity Manager engine.

Components Monitored	Description

userapp_stats	Provides the following information about the JVM statistics for User Application:
	◆ memoryStats: Provides memory related information such as system memory and memory consumed by the JVM.
	◆ runtimeStats: Provides information about the JVM environment, such as start time, uptime, and system properties.
	◆ operatingSystemStats: Provides information about the underlying operating system on which Identity Manager is installed, such as name, version, processor, and architecture.
	◆ classLoaderStats: Provides information about the class loader of the JVM, such as number of loaded classes, unloaded classes, and total number of classes.
	◆ garbageCollectorStats: Provides information about the total number of collections that have occurred and the total collection time in milliseconds.
	Provides the following information for the User Application:
	◆ memory_info: Provides memory related information such as system memory and memory consumed by the JVM.
	◆ thread_info: Provides information about the CPU-intensive threads and returns the list of top threads that cause heavy utilization of the CPU.
	  ◆ threadStateCounts: Provides the current number of live threads.
	◆ cache_info: Provides information about the runtime state of the caching system, and of each of the individual caches for the User Application. OR Provides cache information for the User Application.
	To obtain the User Application information, ensure that the following prerequisites are met:
	◆ Common Setting Advanced Edition package is installed on driver set containing the User Application driver
	  This package contains User Application Provisioning Services URL (`UAProvURL`) and User Application Provisioning Services Administrator (`UAProvAdmin`) GCVs.
	◆ `UAProvURL` and `UAProvAdmin` GCVs have the correct values.
	For more information, see the *NetIQ Identity Manager - Administrator's Guide to the Identity Applications*.

Components Monitored	Description
workflow_stats	Provides the following workflow statistics: ◆ New requests in last 10 seconds, 1 minute, and 5 minutes: Is a count of the number of new requests in the last 10 seconds, 1 minute, and 5 minutes. ◆ In progress workflows: Is a count of the number of workflows in progress. ◆ Terminated workflows: Is a count with the number of terminated workflows. ◆ Failed workflows: Is a count of the number of failed workflows. ◆ Completed workflows in last 10 seconds, 1 minute, and 5 minutes: Is a count of the number of completed workflows in the last 10 seconds, 1 minute, and 5 minutes. ◆ Completed workflows: Is a count of the number of completed workflows. ◆ Total workflows: Is a count of the total number of workflows in regardless of the workflow state. Provides the following JVM statistics for workflow: ◆ memoryStats: Has memory related information such as system memory and memory consumed by the JVM. ◆ runtimeStats: Has information about the JVM environment, such as start time, uptime, and system properties. ◆ operatingSystemStats: Has information about the underlying operating system on which Identity Manager is installed, such as name, version, processor, and architecture. ◆ classLoaderStats: Has information about the class loader of the JVM, such as number of loaded classes, unloaded classes, and total number of classes. ◆ garbageCollectorStats: Has information about the total number of collections that have occurred and the total collection time in milliseconds. Provides the following statistics for workflow: ◆ memory_info: Has memory related information such as system memory and memory consumed by the JVM. ◆ thread_info: Has information about the CPU-intensive threads and returns the list of top threads that cause heavy utilization of the CPU. ◆ threadStateCounts: Has the current number of live threads. ◆ cache_info: Has information about the runtime state of the caching system, and of each of the individual caches for the User Application. OR Provides cache information for the User Application.

To view the monitoring data for all the registered subcomponents of Identity Manager, execute the following ldapsearch command in a command prompt:

```
ldapsearch -h <serverIP> -p <port> -D <user dn> -w <password> -s <search
scope> -b cn=idm,cn=monitor
```

For example:

```
ldapsearch -h 12.345.678.90 -p 389 -D cn=admin,ou=sa,o=system -w <password>
-s sub -b cn=idm,cn=monitor
```

You can perform a base, one-level, or a subtree search. The search returns data from the registered subcomponents in LDAP format using `cn=idm,cn=monitor` as a base of the search. A base search returns all the attributes under the object where you issued the search while a subtree search examines all the objects under the currently searched object. For certain objects, the search is restricted only to the parent object.

Monitoring Identity Manager

The following sections describe the components and the data that can be monitored:

- "Monitoring Job Statistics" on page 145
- "Monitoring JVM Statistics" on page 146
- "Monitoring a Driver Set" on page 148
- "Monitoring User Application Statistics" on page 149
- "Monitoring Workflow Statistics" on page 154

Monitoring Job Statistics

Jobs are administrative functions that can be scheduled for processing at some future date or on a recurring basis. You can monitor the configuration details of jobs at driver set and driver level such as job CN, status of a job, and when a job is scheduled to run.

To view information about the jobs configured in Identity Manager, perform a search on the following entry:

`cn=job_stats,cn=idm,cn=monitor`

To view information about all jobs configured at a driver set level, perform a search on the following entries:

`cn=driverset,cn=job_stats,cn=idm,cn=monitor`

To view information about a specific job configured at a driver set level, perform a search on the following entry:

`cn=<Name of the job>,cn=driverset,cn=job_stats,cn=idm,cn=monitor`

For example, `cn=StatisticsJob,cn=driverset,cn=job_stats,cn=idm,cn=monitor`

To view information about jobs configured for all drivers in a driver set, perform a search on the following entry:

`cn=drivers,cn=job_stats,cn=idm,cn=monitor`

To view information about jobs configured for a driver, perform a search on the following entry:

`cn=<Name of the driver>,cn=drivers,cn=job_stats,cn=idm,cn=monitor`

To view information about a specific job configured for a driver, perform a search on the following entry:

cn=<*CN of the job*>,cn=<*CN of the driver*>,cn=drivers,cn=job_stats,cn=idm,cn=monitor

A sample LDAP search output is below. The search displays nextScheduledRun timestamp in the epoch format.

```
dn: cn=StatisticsJob,cn=DriverSet,cn=JOB_STATS,cn=IDM,cn=Monitor
nextScheduledRun:1485631800
configuration: enabled
status: stopped
containment: DirXML-DriverSet
JobDN: CN=StatisticsJob,CN=driverset1,O=system
objectclass: Top
objectclass: extensibleObject

dn: cn=PermissionOnboarding,cn=Active Directory
Driver,cn=drivers,cn=JOB_STATS,cn=IDM,cn=Monitor
nextScheduledRun: 0
configuration: enabled
status: stopped
containment: DirXML-Driver
JobDN: CN=PermissionOnboarding,CN=Active Directory
Driver,CN=driverset1,O=system
objectclass: Top
objectclass: extensibleObject
```

In this example, nextScheduledRun: 0 means that the job is not scheduled to run.

Monitoring JVM Statistics

You can monitor the current state of the JVM such as memory, thread, runtime, class loader, and garbage collection. This information helps you debug or troubleshoot Java issues such as threading, classpath, and memory buildup. To view this information, perform a search operation on the following entry:

cn=jvm_stats,cn=idm,cn=monitor

You can view your operating system's information such as name, version, architecture, processors, and the load it can take. To monitor this information, perform a search operation on the following entry:

cn=operating_system_stats,cn=jvm_stats,cn=idm,cn=monitor

For information about the current memory allocation of the JVM, perform a search operation on the following entry:

cn=memory_stats,cn=jvm_stats,cn=idm,cn=monitor

To view the class loader of the JVM, perform a search operation on the following entry:

cn=classloader_stats,cn=jvm_stats,cn=idm,cn=monitor

For information about garbage collection of the JVM, perform a search operation on the following entry:

`cn=garbage_collection_stats,cn=jvm_stats,cn=idm,cn=monitor`

For information about thread statistics of the JVM, perform a search operation on the following entry:

`cn=thread_stats,cn=jvm_stats,cn=idm,cn=monitor`

For information about runtime statistics of the JVM, perform a search operation on the following entry:

`cn=runtime_stats,cn=jvm_stats,cn=idm,cn=monitor`

NOTE: For certain objects, Identity Manager restricts the search only to the parent object. For example, `thread_stats` object has only one child node as `thread_info`. Searching under `thread_info` is restricted. This implies that you should not construct a search string such as `cn=thread_info,cn=thread_stats,cn=jvm_stats,cn=idm,cn=monitor`, which will not return you the expected result. Similarly, the `runtime_stats` object contains `system_properties`. Identity Manager does not allow you to separately search for `system_properties`. To monitor these details, you must search for `runtime_stats`.

A sample LDAP search output is below.

```
dn: cn=classloader_stats,cn=jvm_stats,cn=idm,cn=monitor
total_loaded_class_count: 2393
unloaded_class_count: 0
loaded_class_count: 2393
objectclass: Top
objectclass: extensibleObject

dn:
cn=Copy,cn=collectors,cn=garbage_collection_stats,cn=jvm_stats,cn=idm,cn=m
onitor
collection_time: 54 ms
collection_count: 4
objectclass: Top
objectclass: extensibleObject

dn:
cn=MarkSweepCompact,cn=collectors,cn=garbage_collection_stats,cn=jvm_stats
,cn=IDM,cn=Monitor
collection_time: 0 ms
collection_count: 0
objectclass: Top
objectclass: extensibleObject

dn: cn=heap,cn=memory_stats,cn=jvm_stats,cn=idm,cn=monitor
total: 494.94 MB
used: 16.188 MB
comitted: 123.75 MB
initial: 128.00 MB
objectclass: Top
objectclass: extensibleObject

dn: cn=non_heap,cn=memory_stats,cn=jvm_stats,cn=idm,cn=monitor
total: -1.0000 Bytes
```

```
used: 22.064 MB
comitted: 22.688 MB
initial: 2496.0 KB
objectclass: Top
objectclass: extensibleObject

dn: cn=operating_system_stats,cn=jvm_stats,cn=idm,cn=monitor
average_load: 1.0
processors: 1
arch: amd64
version: 3.0.76-0.11-default
name: Linux
objectclass: Top
objectclass: extensibleObject

dn: cn=runtime_stats,cn=jvm_stats,cn=idm,cn=monitor
uptime: 36 days 4 hours 27 minutes 1 seconds 8 milliseconds
start_time: 1478854892218
objectclass: Top
objectclass: extensibleObject

dn: cn=system_properties,cn=runtime_stats,cn=jvm_stats,cn=idm,cn=monitor
sun.boot.class.path:
...
...
```

Monitoring a Driver Set

You can monitor a driver set's information such as driver set DN, the number of drivers the driver set holds, the running state of drivers, and how they are configured to run.

`cn=driverset_stats,cn=idm,cn=monitor`

To monitor information about all the drivers in a driver set, perform a search operation on the following entries:

`cn=drivers,cn=driverset_stats,cn=idm,cn=monitor`

To monitor the statistics of a specific driver such as driver CN, the state of the driver, driver start option, and if the driver is running or down, perform a search operation on the following entries:

`cn=<CN of the driver>,cn=drivers,cn=driverset_stats,cn=idm,cn=monitor`

Identity Manager does not allow you to separately search for `ProcessedOperationsCount` under a driver. To monitor this attribute, perform a subtree search on the driver.

A sample LDAP search output is below.

```
dn: cn=driverset_stats,cn=IDM,cn=Monitor
Startupoption_auto-start_count: 0
Startupoption_manual_count: 10
Startupoption_disabled_count: 1
shutdownpending: 0
starting: 0
running: 0
stopped: 11
numberofdrivers: 11
driversetdn: CN=driverset1,O=system
objectclass: Top
objectclass: extensibleObject

dn: cn=Active Directory
Driver,cn=drivers,cn=driverset_stats,cn=IDM,cn=Monitor
startoption: manual
type: remote
downtime: 6 days 4 hours 27 minutes 2 seconds 9 milliseconds
driver-state: stopped
driverdn: CN=Active Directory Driver,CN=driverset1,O=system
objectclass: Top
objectclass: extensibleObject

dn: cn=publisher,cn=ProcessedOperationsCount,cn=Active Directory
Driver,cn=drivers,cn=driverset_stats,cn=IDM,cn=Monitor
check-password: 14
add: 1
query: 70
modify: 98
objectclass: Top
objectclass: extensibleObject
```

Monitoring User Application Statistics

Using the monitoring feature you can execute search operations using specific entries to monitor the condition of the JVM and get statistics for elements such as memory usage, thread activity, runtime statistics, class loader status, garbage collection, and operating system. You can use this data for debugging or addressing Java-related issues.

IMPORTANT: To receive the cluster setup JVM statistics for user application, you must add the following property in all the cluster nodes in the `ism.configuration.properties` file:
`com.netiq.cluster.internalNodes = {\"nodes\":[{\"ip\":\"<serverIP>\",\"port\":\"<port>\"},{\"ip\":\"<serverIP>\",\"port\":\"<port>\"}]}`.

For example, `com.netiq.cluster.internalNodes = {\"nodes\":[{\"ip\":\"12.345.678.90\",\"port\":\"8543\"},{\"ip\":\"56.789.123.50\",\"port\":\"8543\"}]}`.

You must add the property in the default locations:

- Linux: `/opt/netiq/idm/apps/tomcat/conf/ism-configuration.properties`
- Windows: `C:\NetIQ\IDM\apps\tomcat\conf\ism-configuration.properties`

To view the JVM statistics for user applications, perform a search using the following entry:

`cn=jvmstats_info,cn=userapp_stats,cn=idm,cn=monitor`

You can monitor information about the health of the User Application such as currently running threads, memory consumption, and cache information. Note that `memory_info`, `thread_info`, and `cache_info` does not provide information for clustered environment.

To view the health statistics of the User Application, perform a search on the following entry:

`cn=userapp_stats,cn=idm,cn=monitor`

To monitor the system memory and memory consumed by the JVM, perform a search on the following entry:

`cn=memory_info,cn=userapp_stats,cn=idm,cn=monitor`

NOTE: `memory_info` contains several child objects such as `jvmmemoryusage` and `jvmmemorypool`. However, Identity Manager restricts the search to the parent object (memory_info)only.

To monitor the threads that cause heavy utilization of the CPU, perform a search with the following entry:

`cn=thread_info,cn=userapp_stats,cn=idm,cn=monitor`

To view the current number of live threads, perform a search on the following entry:

`cn=thread_count,cn=thread_info,cn=userapp_stats,cn=idm,cn=monitor`

To view the cache information, perform a search on the following entry:

`cn=cache_info,cn=userapp_stats,cn=idm,cn=monitor`

A sample output looks like this:

```
dn:cn=62614,cn=classLoaderStats,cn=Node_1,cn=nodesStats,cn=jvmstats_info,c
n=u
 serapp_stats,cn=IDM,cn=Monitor
unloadedClassCount: 25
totalLoadedClassCount: 62639
objectclass: Top
objectclass: extensibleObject

# Node_1, nodesStats, jvmstats_info, userapp_stats, IDM, Monitor
dn:
cn=Node_1,cn=nodesStats,cn=jvmstats_info,cn=userapp_stats,cn=IDM,cn=Monito
 r
currentTimeStamp: 2024-04-05 07:17:15.054
objectclass: Top
objectclass: extensibleObject

# G1 Old Generation, garbageCollectorStats, Node_1, nodesStats,
jvmstats_info,
 userapp_stats, IDM, Monitor
dn: cn=G1 Old
Generation,cn=garbageCollectorStats,cn=Node_1,cn=nodesStats,cn=j
```

```
vmstats_info,cn=userapp_stats,cn=IDM,cn=Monitor
collectionTime: 0
collectionCount: 0
objectclass: Top
objectclass: extensibleObject

# G1 Young Generation, garbageCollectorStats, Node_1, nodesStats,
jvmstats_info
 , userapp_stats, IDM, Monitor
dn: cn=G1 Young
Generation,cn=garbageCollectorStats,cn=Node_1,cn=nodesStats,cn
 =jvmstats_info,cn=userapp_stats,cn=IDM,cn=Monitor
collectionTime: 2873
collectionCount: 59
objectclass: Top
objectclass: extensibleObject

# Node_1, nodesStats, jvmstats_info, userapp_stats, IDM, Monitor
dn:
cn=Node_1,cn=nodesStats,cn=jvmstats_info,cn=userapp_stats,cn=IDM,cn=Monito
 r
ipAddressAndPort: 10.71.39.111:-1
objectclass: Top
objectclass: extensibleObject

# memoryStats, Node_1, nodesStats, jvmstats_info, userapp_stats, IDM,
Monitor
dn:
cn=memoryStats,cn=Node_1,cn=nodesStats,cn=jvmstats_info,cn=userapp_stats,c
 n=IDM,cn=Monitor
commitedVirtualMemory: 1818044
objectclass: Top
objectclass: extensibleObject

# CodeHeap 'non-nmethods', jvmMemoryPool, memoryStats, Node_1, nodesStats,
jvms
 tats_info, userapp_stats, IDM, Monitor
dn: cn=CodeHeap 'non-
nmethods',cn=jvmMemoryPool,cn=memoryStats,cn=Node_1,cn=no
 desStats,cn=jvmstats_info,cn=userapp_stats,cn=IDM,cn=Monitor
freeMemory: 3902
usedMemory: 1794
totalMemory: 5696
objectclass: Top
objectclass: extensibleObject

# CodeHeap 'non-profiled nmethods', jvmMemoryPool, memoryStats, Node_1,
nodesSt
 ats, jvmstats_info, userapp_stats, IDM, Monitor
dn: cn=CodeHeap 'non-profiled
nmethods',cn=jvmMemoryPool,cn=memoryStats,cn=Nod
 e_1,cn=nodesStats,cn=jvmstats_info,cn=userapp_stats,cn=IDM,cn=Monitor
freeMemory: 90616
usedMemory: 29448
totalMemory: 120064
```

```
objectclass: Top
objectclass: extensibleObject

# CodeHeap 'profiled nmethods', jvmMemoryPool, memoryStats, Node_1,
nodesStats,
  jvmstats_info, userapp_stats, IDM, Monitor
dn: cn=CodeHeap 'profiled
nmethods',cn=jvmMemoryPool,cn=memoryStats,cn=Node_1,
 cn=nodesStats,cn=jvmstats_info,cn=userapp_stats,cn=IDM,cn=Monitor
freeMemory: 35801
usedMemory: 84199
totalMemory: 120000
objectclass: Top
objectclass: extensibleObject

# Compressed Class Space, jvmMemoryPool, memoryStats, Node_1, nodesStats,
jvmst
 ats_info, userapp_stats, IDM, Monitor
dn: cn=Compressed Class
Space,cn=jvmMemoryPool,cn=memoryStats,cn=Node_1,cn=nod
 esStats,cn=jvmstats_info,cn=userapp_stats,cn=IDM,cn=Monitor
freeMemory: 1003995
usedMemory: 44581
totalMemory: 1048576
objectclass: Top
objectclass: extensibleObject

# G1 Eden Space, jvmMemoryPool, memoryStats, Node_1, nodesStats,
jvmstats_info,
  userapp_stats, IDM, Monitor
dn: cn=G1 Eden
Space,cn=jvmMemoryPool,cn=memoryStats,cn=Node_1,cn=nodesStats,c
 n=jvmstats_info,cn=userapp_stats,cn=IDM,cn=Monitor
freeMemory: 0
usedMemory: 52224
totalMemory: 0
objectclass: Top
objectclass: extensibleObject

# G1 Old Gen, jvmMemoryPool, memoryStats, Node_1, nodesStats,
jvmstats_info, us
 erapp_stats, IDM, Monitor
dn: cn=G1 Old
Gen,cn=jvmMemoryPool,cn=memoryStats,cn=Node_1,cn=nodesStats,cn=j
 vmstats_info,cn=userapp_stats,cn=IDM,cn=Monitor
freeMemory: 748903
usedMemory: 299673
totalMemory: 1048576
objectclass: Top
objectclass: extensibleObject

# G1 Survivor Space, jvmMemoryPool, memoryStats, Node_1, nodesStats,
jvmstats_i
 nfo, userapp_stats, IDM, Monitor
dn: cn=G1 Survivor
```

```
Space,cn=jvmMemoryPool,cn=memoryStats,cn=Node_1,cn=nodesSta
 ts,cn=jvmstats_info,cn=userapp_stats,cn=IDM,cn=Monitor
freeMemory: 0
usedMemory: 7168
totalMemory: 0
objectclass: Top
objectclass: extensibleObject

# Metaspace, jvmMemoryPool, memoryStats, Node_1, nodesStats, jvmstats_info,
use
 rapp_stats, IDM, Monitor
dn:
cn=Metaspace,cn=jvmMemoryPool,cn=memoryStats,cn=Node_1,cn=nodesStats,cn=jv
 mstats_info,cn=userapp_stats,cn=IDM,cn=Monitor
freeMemory: 0
usedMemory: 376517
totalMemory: 0
objectclass: Top
objectclass: extensibleObject

# HeapMemoryUsage, jvmMemoryUsage, memoryStats, Node_1, nodesStats,
jvmstats_in
 fo, userapp_stats, IDM, Monitor
dn:
cn=HeapMemoryUsage,cn=jvmMemoryUsage,cn=memoryStats,cn=Node_1,cn=nodesStat
 s,cn=jvmstats_info,cn=userapp_stats,cn=IDM,cn=Monitor
freeMemory: 689511
usedMemory: 359065
totalMemory: 1048576
objectclass: Top
objectclass: extensibleObject

# NonHeapMemoryUsage, jvmMemoryUsage, memoryStats, Node_1, nodesStats,
jvmstats
 _info, userapp_stats, IDM, Monitor
dn:
cn=NonHeapMemoryUsage,cn=jvmMemoryUsage,cn=memoryStats,cn=Node_1,cn=nodesS
 tats,cn=jvmstats_info,cn=userapp_stats,cn=IDM,cn=Monitor
freeMemory: 0
usedMemory: 536541
totalMemory: 0
objectclass: Top
objectclass: extensibleObject

# physicalMemory, systemMemoryUsage, memoryStats, Node_1, nodesStats,
jvmstats_
 info, userapp_stats, IDM, Monitor
dn:
cn=physicalMemory,cn=systemMemoryUsage,cn=memoryStats,cn=Node_1,cn=nodesSt
 ats,cn=jvmstats_info,cn=userapp_stats,cn=IDM,cn=Monitor
freeMemory: 3225320
usedMemory: 5162764
totalMemory: 8388084
objectclass: Top
objectclass: extensibleObject
```

```
# swapMemory, systemMemoryUsage, memoryStats, Node_1, nodesStats,
jvmstats_info
 , userapp_stats, IDM, Monitor
dn:
cn=swapMemory,cn=systemMemoryUsage,cn=memoryStats,cn=Node_1,cn=nodesStats,
 cn=jvmstats_info,cn=userapp_stats,cn=IDM,cn=Monitor
freeMemory: 3204888
usedMemory: 6493916
totalMemory: 9698804
objectclass: Top
objectclass: extensibleObject

# Windows Server 2022, operatingSystemStats, Node_1, nodesStats,
jvmstats_info,
  userapp_stats, IDM, Monitor
dn: cn=Windows Server
2022,cn=operatingSystemStats,cn=Node_1,cn=nodesStats,cn=
 jvmstats_info,cn=userapp_stats,cn=IDM,cn=Monitor
systemAverageLoad: -1
arch: amd64
processorsCount: 2
version: 10.0
objectclass: Top
objectclass: extensibleObject

# 20381464, runtimeStats, Node_1, nodesStats, jvmstats_info, userapp_stats,
IDM
 , Monitor
dn:
cn=20381464,cn=runtimeStats,cn=Node_1,cn=nodesStats,cn=jvmstats_info,cn=us
 erapp_stats,cn=IDM,cn=Monitor
startTime: 1712261253
objectclass: Top
objectclass: extensibleObject

# search result
search: 2
result: 0 Success

# numResponses: 21
# numEntries: 20
```

Monitoring Workflow Statistics

With the monitoring feature, you can monitor the workflows and the condition of the JVM and gather information about various aspects using specific search entries. You can get statistics on JVM's memory usage, thread activity, runtime details, class loader status, garbage collection, and the operating system. Additionally, you can get information about the overall count of workflows, those in progress, completed, failed, terminated, or requested within the last 10 seconds, 1 minute, and 5 minutes.

To know the workflow statistics, perform a search using the following entry:

```
cn=workflowstatistics_info,cn=workflow_stats,cn=idm,cn=monitor
```

The following is a sample workflow statistics search output:

```
dn:
cn=workflowstatistics_info,cn=workflow_stats,cn=IDM,cn=Monitor
currentTimeStamp: 2024-04-01 08:09:37.793
newRequestsInLast10Secs: 0
newRequestsInLast1Min: 1
newRequestsInLast5Mins: 1
inProgressWorkflows: 1
terminatedWorkflows: 1
failedWorkflows: 0
completedWorkflowsInLast10Secs: 0
completedWorkflowsInLast1Min: 0
completedWorkflowsInLast5Mins: 1
completedWorkflows: 2
totalWorkflows: 4
objectclass: Top
objectclass: extensibleObject

# search result
search: 2
result: 0 Success

# numResponses: 88
# numEntries: 87
```

IMPORTANT: To receive the cluster setup JVM statistics for workflow, you must add the following property in all the cluster nodes in the `ism.configuration.properties` file:
`com.netiq.cluster.internalNodes = {\"nodes\":[{\"ip\":\"<serverIP>\",\"port\":\"<port>\"},{\"ip\":\"<serverIP>\",\"port\":\"<port>\"}]}`.

For example, `com.netiq.cluster.internalNodes = {\"nodes\":[{\"ip\":\"12.345.678.90\",\"port\":\"8543\"},{\"ip\":\"56.789.123.50\",\"port\":\"8543\"}]}`.

You must add the property in the default locations:

- Linux: `/opt/netiq/idm/apps/tomcat/conf/ism-configuration.properties`
- Windows: `C:\NetIQ\IDM\apps\tomcat\conf\ism-configuration.properties`

To know the JVM statistics for workflows, perform a search using the following entry:

```
cn=jvmstats_info,cn=workflow_stats,cn=IDM,cn=Monitor
```

To monitor the system memory and memory consumed by the JVM, perform a search on the following entry. Note that `memory_info`, `thread_info`, and `cache_info` does not provide information for clustered environment.

```
cn=memory_info,cn=workflow_stats,cn=idm,cn=monitor
```

NOTE: `memory_info` contains several child objects such as `jvmmemoryusage` and `jvmmemorypool`. However, Identity Manager restricts the search to the parent object (memory_info)only.

To monitor the threads that cause heavy utilization of the CPU, perform a search with the following entry:

```
cn=thread_info,cn=workflow_stats,cn=idm,cn=monitor
```

To view the current number of live threads, perform a search on the following entry:

```
cn=thread_count,cn=thread_info,cn=userapp_stats,cn=idm,cn=monitor
```

To view the cache information, perform a search on the following entry:

```
cn=cache_info,cn=workflow_stats,cn=idm,cn=monitor
```

The following is a sample JVM statistics search output for workflow:

```
dn:
cn=62618,cn=classLoaderStats,cn=Node_1,cn=nodesStats,cn=jvmstats_info,cn=w
 orkflow_stats,cn=IDM,cn=Monitor
unloadedClassCount: 25
totalLoadedClassCount: 62643
objectclass: Top
objectclass: extensibleObject

# Node_1, nodesStats, jvmstats_info, workflow_stats, IDM, Monitor
dn:
cn=Node_1,cn=nodesStats,cn=jvmstats_info,cn=workflow_stats,cn=IDM,cn=Monit
 or
currentTimeStamp: 2024-04-05 07:17:56.996
objectclass: Top
objectclass: extensibleObject

# G1 Old Generation, garbageCollectorStats, Node_1, nodesStats,
jvmstats_info,
 workflow_stats, IDM, Monitor
dn: cn=G1 Old
Generation,cn=garbageCollectorStats,cn=Node_1,cn=nodesStats,cn=j
 vmstats_info,cn=workflow_stats,cn=IDM,cn=Monitor
collectionTime: 0
collectionCount: 0
objectclass: Top
objectclass: extensibleObject

# G1 Young Generation, garbageCollectorStats, Node_1, nodesStats,
jvmstats_info
 , workflow_stats, IDM, Monitor
dn: cn=G1 Young
Generation,cn=garbageCollectorStats,cn=Node_1,cn=nodesStats,cn
 =jvmstats_info,cn=workflow_stats,cn=IDM,cn=Monitor
collectionTime: 2873
collectionCount: 59
objectclass: Top
objectclass: extensibleObject
```

```
# Node_1, nodesStats, jvmstats_info, workflow_stats, IDM, Monitor
dn:
cn=Node_1,cn=nodesStats,cn=jvmstats_info,cn=workflow_stats,cn=IDM,cn=Monit
 or
ipAddressAndPort: 10.71.39.111:-1
objectclass: Top
objectclass: extensibleObject

# memoryStats, Node_1, nodesStats, jvmstats_info, workflow_stats, IDM,
Monitor
dn:
cn=memoryStats,cn=Node_1,cn=nodesStats,cn=jvmstats_info,cn=workflow_stats,
 cn=IDM,cn=Monitor
commitedVirtualMemory: 1818264
objectclass: Top
objectclass: extensibleObject

# CodeHeap 'non-nmethods', jvmMemoryPool, memoryStats, Node_1, nodesStats,
jvms
 tats_info, workflow_stats, IDM, Monitor
dn: cn=CodeHeap 'non-
nmethods',cn=jvmMemoryPool,cn=memoryStats,cn=Node_1,cn=no
 desStats,cn=jvmstats_info,cn=workflow_stats,cn=IDM,cn=Monitor
freeMemory: 3902
usedMemory: 1794
totalMemory: 5696
objectclass: Top
objectclass: extensibleObject

# CodeHeap 'non-profiled nmethods', jvmMemoryPool, memoryStats, Node_1,
nodesSt
 ats, jvmstats_info, workflow_stats, IDM, Monitor
dn: cn=CodeHeap 'non-profiled
nmethods',cn=jvmMemoryPool,cn=memoryStats,cn=Nod
 e_1,cn=nodesStats,cn=jvmstats_info,cn=workflow_stats,cn=IDM,cn=Monitor
freeMemory: 90602
usedMemory: 29462
totalMemory: 120064
objectclass: Top
objectclass: extensibleObject

# CodeHeap 'profiled nmethods', jvmMemoryPool, memoryStats, Node_1,
nodesStats,
  jvmstats_info, workflow_stats, IDM, Monitor
dn: cn=CodeHeap 'profiled
nmethods',cn=jvmMemoryPool,cn=memoryStats,cn=Node_1,
 cn=nodesStats,cn=jvmstats_info,cn=workflow_stats,cn=IDM,cn=Monitor
freeMemory: 35753
usedMemory: 84247
totalMemory: 120000
objectclass: Top
objectclass: extensibleObject

# Compressed Class Space, jvmMemoryPool, memoryStats, Node_1, nodesStats,
```

```
jvmst
 ats_info, workflow_stats, IDM, Monitor
dn: cn=Compressed Class
Space,cn=jvmMemoryPool,cn=memoryStats,cn=Node_1,cn=nod
 esStats,cn=jvmstats_info,cn=workflow_stats,cn=IDM,cn=Monitor
freeMemory: 1003993
usedMemory: 44583
totalMemory: 1048576
objectclass: Top
objectclass: extensibleObject

# G1 Eden Space, jvmMemoryPool, memoryStats, Node_1, nodesStats,
jvmstats_info,
  workflow_stats, IDM, Monitor
dn: cn=G1 Eden
Space,cn=jvmMemoryPool,cn=memoryStats,cn=Node_1,cn=nodesStats,c
 n=jvmstats_info,cn=workflow_stats,cn=IDM,cn=Monitor
freeMemory: 0
usedMemory: 70656
totalMemory: 0
objectclass: Top
objectclass: extensibleObject

# G1 Old Gen, jvmMemoryPool, memoryStats, Node_1, nodesStats,
jvmstats_info, wo
 rkflow_stats, IDM, Monitor
dn: cn=G1 Old
Gen,cn=jvmMemoryPool,cn=memoryStats,cn=Node_1,cn=nodesStats,cn=j
 vmstats_info,cn=workflow_stats,cn=IDM,cn=Monitor
freeMemory: 747879
usedMemory: 300697
totalMemory: 1048576
objectclass: Top
objectclass: extensibleObject

# G1 Survivor Space, jvmMemoryPool, memoryStats, Node_1, nodesStats,
jvmstats_i
 nfo, workflow_stats, IDM, Monitor
dn: cn=G1 Survivor
Space,cn=jvmMemoryPool,cn=memoryStats,cn=Node_1,cn=nodesSta
 ts,cn=jvmstats_info,cn=workflow_stats,cn=IDM,cn=Monitor
freeMemory: 0
usedMemory: 7168
totalMemory: 0
objectclass: Top
objectclass: extensibleObject

# Metaspace, jvmMemoryPool, memoryStats, Node_1, nodesStats, jvmstats_info,
wor
 kflow_stats, IDM, Monitor
dn:
cn=Metaspace,cn=jvmMemoryPool,cn=memoryStats,cn=Node_1,cn=nodesStats,cn=jv
 mstats_info,cn=workflow_stats,cn=IDM,cn=Monitor
freeMemory: 0
usedMemory: 376564
```

```
totalMemory: 0
objectclass: Top
objectclass: extensibleObject

# HeapMemoryUsage, jvmMemoryUsage, memoryStats, Node_1, nodesStats,
jvmstats_in
 fo, workflow_stats, IDM, Monitor
dn:
cn=HeapMemoryUsage,cn=jvmMemoryUsage,cn=memoryStats,cn=Node_1,cn=nodesStat
 s,cn=jvmstats_info,cn=workflow_stats,cn=IDM,cn=Monitor
freeMemory: 670055
usedMemory: 378521
totalMemory: 1048576
objectclass: Top
objectclass: extensibleObject

# NonHeapMemoryUsage, jvmMemoryUsage, memoryStats, Node_1, nodesStats,
jvmstats
 _info, workflow_stats, IDM, Monitor
dn:
cn=NonHeapMemoryUsage,cn=jvmMemoryUsage,cn=memoryStats,cn=Node_1,cn=nodesS
 tats,cn=jvmstats_info,cn=workflow_stats,cn=IDM,cn=Monitor
freeMemory: 0
usedMemory: 536652
totalMemory: 0
objectclass: Top
objectclass: extensibleObject

# physicalMemory, systemMemoryUsage, memoryStats, Node_1, nodesStats,
jvmstats_
 info, workflow_stats, IDM, Monitor
dn:
cn=physicalMemory,cn=systemMemoryUsage,cn=memoryStats,cn=Node_1,cn=nodesSt
 ats,cn=jvmstats_info,cn=workflow_stats,cn=IDM,cn=Monitor
freeMemory: 3225340
usedMemory: 5162744
totalMemory: 8388084
objectclass: Top
objectclass: extensibleObject

# swapMemory, systemMemoryUsage, memoryStats, Node_1, nodesStats,
jvmstats_info
 , workflow_stats, IDM, Monitor
dn:
cn=swapMemory,cn=systemMemoryUsage,cn=memoryStats,cn=Node_1,cn=nodesStats,
 cn=jvmstats_info,cn=workflow_stats,cn=IDM,cn=Monitor
freeMemory: 3203372
usedMemory: 6495432
totalMemory: 9698804
objectclass: Top
objectclass: extensibleObject

# Windows Server 2022, operatingSystemStats, Node_1, nodesStats,
jvmstats_info,
  workflow_stats, IDM, Monitor
```

```
dn: cn=Windows Server
2022,cn=operatingSystemStats,cn=Node_1,cn=nodesStats,cn=
 jvmstats_info,cn=workflow_stats,cn=IDM,cn=Monitor
systemAverageLoad: -1
arch: amd64
processorsCount: 2
version: 10.0
objectclass: Top
objectclass: extensibleObject

# 20423407, runtimeStats, Node_1, nodesStats, jvmstats_info,
workflow_stats, ID
 M, Monitor
dn:
cn=20423407,cn=runtimeStats,cn=Node_1,cn=nodesStats,cn=jvmstats_info,cn=wo
 rkflow_stats,cn=IDM,cn=Monitor
startTime: 1712261253
objectclass: Top
objectclass: extensibleObject

# search result
search: 2
result: 0 Success

# numResponses: 21
# numEntries: 20
```

20 Improving Driver Performance Using Subscriber Service Channel

Identity Manager processes events in a sequential way. When an out of band query is issued, an Identity Manager driver starts processing the query after it finishes processing the current event. After the query is processed, the driver resumes normal operations. Some examples of out of band queries are code map refresh, data collection, and queries triggered from `dxcmd`. Processing out of band queries pauses the normal flow of events until the driver finishes processing the query that in turn impacts the driver's performance. To improve the performance, Identity Manager provides the Subscriber Service channel for handling such queries. This channel separately processes these queries without interrupting the flow of other events.

The Subscriber Service channel does not process all the polices. The following policies are processed:

- Command transformation
- Schema mapping

Currently, the following drivers support this channel: Active Directory, Multi-Domain Active Directory, JDBC, and JDBC Fan-Out.

NOTE: Identity Manager drivers that do not support the Subscriber Service channel process out of band queries through the Subscriber channel.

To enable the Subscriber Service channel for a supported Identity Manager driver, see "Configuring the Subscriber Service Channel" on page 161.

Prerequisites

- Identity Manager 4.6 or later
- Remote Loader 4.7 or later
- Driver shim enabled with Subscriber Service channel

Configuring the Subscriber Service Channel

You can change the default behavior of the Subscriber Service channel for a driver by using **Enable Subscriber Service Channel** Engine Control Value (ECV) through Designer or Identity Console. For more information about the ECV, see "Engine Control Values" on page 240.

To change the ECV in Designer:

1. In Modeler, right-click the driver line.
2. Select **Properties > Engine Control Values**.
3. Click the tooltip icon to the right of **Engine Controls for Server**.

If a server is associated with the Identity Vault, and if you are authenticated, the engine control values display in the large pane.

4 Change the value for **Enable Subscriber Service Channel**.

5 Click **OK**.

6 For the change to take effect, deploy the driver to the live Identity Vault.

To change the ECV in Identity Console:

1 Log in to the instance of Identity Console that manages your Identity Vault.

2 In the navigation frame, select **Identity Manager**.

3 Select **Identity Manager Overview**.

4 Use the search page to display the Identity Manager Overview for the driver set that contains your driver.

5 Click the round status indicator in the upper right corner of the driver icon.

6 Select **Edit Properties > Engine Control Values**.

7 Change the value for **Enable Subscriber Service Channel**.

8 Click **OK**, then click **Apply**.

9 For the change to take effect, restart the driver.

Subscriber Service Channel Support Compatibility

The ECV that controls the Subscriber Service Channel support (**Enable Subscriber Service Channel**) was introduced in Identity Manager 4.6. Note that Subscriber Service Channel will work only when the driver that you are loading into the engine is also supported with this feature. For example, JDBC driver. To support this channel with the Remote Loader, you must upgrade to Identity Manager 4.7.

Remote Interface Shim (Engine)	Remote Loader	Driver	Enable Service Channel
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	Yes/No	No
No	Yes/No	Yes/No	No

Tracing the Service Channel

You can monitor Subscriber Service channel query processing using Trace. When the channel is enabled on a driver, Identity Manager prints the trace output similar to the following:

```
Oracle FanOut Driver SST:Injecting User Agent XDS command document into
Subscriber service channel.
[10/27/16 01:15:05.940]Oracle FanOut Driver SST:Applying command
transformation policies.
[10/27/16 01:15:05.940]Oracle FanOut Driver SST:Applying policy:
%+C%14CNETQJFOMNSIS-sub-ctp-HandleMSGWQueries%-C.
[10/27/16 01:15:05.940]Oracle FanOut Driver SST:Applying policy:
%+C%14CNETQFOUTCOMM-sub-ctp-UpdateAddDocWithCurrentStateOfUser%-C.
[10/27/16 01:15:05.940]Oracle FanOut Driver SST:Applying policy:
%+C%14CNETQFOUTCOMM-sub-ctp-TransformAssociationsinAddEvent%-C.
[10/27/16 01:15:05.941]Oracle FanOut Driver SST:Applying policy:
%+C%14CNETQJFOENTIS-sub-ctp-FanOutEntiImpl-I%-C.
```

In this trace output, the Subscriber Service channel is depicted by **SST**. The channel is traced at the same level that is configured for the driver for which the channel is enabled.

Identity Manager also provides a separate trace file for monitoring the Subscriber Service channel operations. For example, if the name of the driver trace file is `driver.log`, another file is created as `driver_svc.log`. This file captures entries for out of band queries.

Trace messages are tagged based on the execution of Publisher or Subscriber threads. For example, when a Subscriber Service channel thread executes, the thread is traced with SST tag.

There is no change in the Remote Loader trace.

21 Viewing Identity Manager Processes

To view Identity Manager processing events, use Identity Manager trace functionality. You use trace only during testing and troubleshooting Identity Manager. Running trace while the drivers are in production increases the load on the Identity Manager server and can cause events to process very slowly.

To see Identity Manager processes in trace, you add values to the driver set and the drivers. You can do this in Designer and Identity Console. For a detailed information about how tracing works in Identity Manager, see Appendix C, "Understanding Identity Manager Trace," on page 249.

Setting Permission for Monitoring Trace Files

In this release, the Identity Manager install program creates a new group named `idvadmin` on Linux. The permission for the trace file is set to 640. To enable a user other than an eDirectory owner to monitor the trace files, you must add that user to the `idvadmin` group.

Working with is-sensitive Attribute

The DirXML Script and XSLT methods of creating policies apply `is-sensitive` attribute on the XDS nodes to hide values of sensitive attributes such as passwords in the trace file. When this attribute is used with a driver, Identity Manager prints the trace output similar to the following:

```
Driver : Applying policy: %+C%14Cis-sensetive%-C.
Driver : Applying to add #1.
Driver :    Evaluating selection criteria for rule 'is-senesitive '.
Driver :      (if-attr 'secret' available) = TRUE.
Driver :    Rule selected.
Driver :    Applying rule 'is-senesitive '.
Driver :      Action: do-set-xml-attr("is-sensitive","*[@attr-
name='secret']","true").
Driver :        arg-string("true")
Driver :          token-text("true")
Driver :          Arg Value: "true".
Driver :Policy returned:
Driver :
<nds dtdversion="4.0" ndsversion="8.x">
  <source>
    <product version="4.6.0.0">DirXML</product>
    <contact>NetIQ Corporation</contact>
  </source>
  <input>
    <add class-name="User" src-dn="data/users/user1">
      <add-attr attr-name="secret" is-sensitive="true"><!-- content
suppressed -->
      </add-attr>
    </add>
  </input>
</nds>
```

22 Editing Driver Configuration Files

Driver configuration files are replaced with packages. You can still use driver configuration files, but all updated driver information is contained in packages. For more information, see "Understanding Packages" in the *NetIQ Designer for Identity Manager Administration Guide*.

You must have a good knowledge of XML to use the information in this section. This information allows you to add custom prompts to drivers you have created.

Variables in a Driver Configuration File

Several node types are defined for the driver configuration files. The following is a list of actions that the Identity Manager engine supports:

- Prompting once for a value that is used repeatedly throughout a single driver configuration file.
- Prompting once for a value that is used across multiple driver configuration files, as part of the Import Drivers Wizard.
- Allowing the user to select a value from a drop-down list of values.
- Global modification of the driver configuration files according to a contained XSL style sheet.
- Built-in variables that can be referenced without declaring them to access information about the driver and its environment. For example, tree name, driver set name, driver set DN, server name, server DN, driver name, and driver DN.
- The ability to layer prompts. It is possible to ask the user multiple sets of questions, with the second and later sets being controlled by the user's responses to prior sets of questions. For more information, see "Flexible Prompting in a Driver Configuration File" on page 171.

The primary new node types are:

- **variable-decl:** Allows you to define driver configuration variables that are prompted for and placed into a driver configuration file during its import. Multiple `variable-decl` blocks can be used to define a layered set of prompts. For more information, see "Flexible Prompting in a Driver Configuration File" on page 171.
- **variable-ref:** Used to reference a variable defined in a `variable-decl` within your driver configuration files.
- **xsl-modify:** Used to globally modify the driver configuration file after all variables and prompts have been resolved. The contents of this node are extracted and used as an XSL style sheet that is applied to the patched driver configuration file.

To view the driver configuration file XML extensions, see DriverConfigXMLExtension.txt (../samples/DriverConfigXMLExtension.txt).

In addition, be aware of the following:

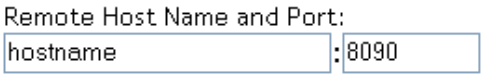
- "General Notes" on page 168
- "Import Driver Notes" on page 170

General Notes

Review the following general notes:

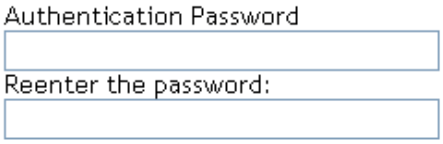
- A `variable-decl` can contain `text-var` but not `node-var`. It can contain `variable-refs` as long as the order they are resolved is taken into account.
- If a `variable-decl` contains an optional prompt attribute and an optional prompt-type attribute and does not contain an optional `browse="yes"` attribute setting, the prompt-type is treated as follows:
  - A prompt-type of `"ipa"` results in two edit fields. See Figure 22-1 for an example. The value the user specifies for the first part is appended by a colon (:) and the value the user specifies for the second part in the value is rendered by the variable.

    Figure 22-1 *Two Edit Fields*

  - A prompt-type of `"password"` results in two password edit fields. See Figure 22-2 for an example. The first prompt is for the actual password, and the second prompt is used to verify that the password specified in the first field is correct. The value rendered by the variable reference is the password.

    Figure 22-2 *Two Password Fields*

  - A prompt-type of `"hidden"` results in a field that is not displayed, but is checked to make sure a previous condition is met before proceeding to the next screen.
  - Any other prompt-type value is ignored.
- If a `variable-decl` contains an optional description attribute in addition to a prompt attribute, the description appears in the UI along with the prompt. The purpose of the description attribute is to allow a complete description of what is being asked for along with a simple prompt.

  For example:

  ```
  <text-var
          var-name="eProv.Company"
          prompt="Company name:" description="Please enter the name of
  your company. This must be the same name as you entered during the
  initial installation."
          browse="no">
        Novell
      </text-var>
  ```

  Note the differences between the `prompt` and the `description`.

If a `variable-decl` contains an optional description attribute and an optional highlight attribute, the highlight attribute is handled as follows:

- If the highlight is not two characters in length, it is ignored.
- If the highlight is two characters in length, all occurrences of the first character are preceded with HTML tags to turn on highlighting and all occurrences of the second character are followed by HTML tags to turn off highlighting.

  For example:

  ```
  <text-var
              var-name="foo"
              prompt="Foo:"
             description="Please enter some foo.  Format:  [foo looks
  like this]">
            Bar
          </text-var>
  ```

  When the description appears, [foo looks like this] appears and is highlighted.

- If a `variable-decl` contains a `browse="yes"` attribute, it is assumed to supply a DN and is formatted in slash format by default when applied to the driver configuration file.

  This is assumed to be more generally useful for driver writers and can be overridden on a per reference basis by adding a `dn-format="dot"` attribute to `variable-ref` nodes that reference it.

- If a `variable-ref` is to `text-var` with a `prompt-type="ipa"` attribute, a `part="..."` attribute can be included in the `variable-ref`. Supported parts are `"ipa"` and `"port"`. If `part="ipa"` is specified, only the IP address portion of the variable's value is returned. If `part="port"` is specified, only the port portion of the variable's value is returned. Any other setting is ignored and the variable's entire value is returned.
- A `dn-format` attribute on a `variable-ref` that does not have `browse="yes"` specified in its `variable-decl` causes that variable to be treated as though it supplies a DN. The DN is rendered in the `dn-format` specified.
- The supported values for the `dn-format` attribute are `"dot"` and `"slash"`. Any other value is treated as `"slash"` without an error being generated.
- The built-in defined variables are:
  - System.TreeName
  - System.DSetDN
  - System.DSetName
  - System.DriverDN
  - System.DriverName
  - System.ServerDN
  - System.ServerName
- Built-in variables can be overridden. If you include a `variable-decl` for a variable named with one of the built-in variable names, your definition overrides the built-in variable of the same name.

  This is implemented after all variable declarations have been processed (prompting, ...). Just before the code begins applying values, it walks the variables and defines all the built-ins that haven't otherwise been defined.

- The built-in variables that provide a DN can include a `dn-format` attribute in the variable-ref to control the format the DN is rendered in. By default, these are rendered in slash format.
- A `node-var` and a `text-var` cannot be named the same thing. They use the same namespace.
- If a `variable-ref` references a `node-var` and contains an `attr-name` attribute, the XSL string value of the `node-var` is stored in as the named attribute on the parent node of the `variable-ref`. The `node-var` used in this manner can have a `node-name` attribute of `"#text"`, which removes the requirement of having an `attr-name` attribute on the `node-var`.

  A `node-var` with a node-name of `"#text"` can be referenced only in this manner. Any other reference causes an error when the driver configuration file is imported.
- At patch time after the user has responded to the prompts but before the XML is actually imported. patching is done in the following order:
  1. The `text-var` `variable-refs` are processed.
  2. The `node-var` `variable-refs` are processed.
  3. The `xsl-modify` commands are processed.
  4. The `ds-object` commands are processed.
     - Patching is performed in the `variable-decl` so that by the time the `node-var` commands are patched, all the `text-var` commands contained in them have been resolved.
     - The `node-var` commands cannot contain `node-var variable-ref`.

Import Driver Notes

Review the following import driver notes:

- The order in which the selected driver configuration files are processed is not defined and no order can be assumed.
- For `variable-decl` commands:
  - Commands from selected drivers are carried forward from driver to driver.
  - The first one wins.
    - The first driver encountered that defines a variable foo has its variable foo used throughout all remaining driver configuration files. Care must be taken to coordinate this between drivers.
    - A variable foo that is used in multiple driver configuration files is prompted for only once, with the first driver configuration file encountered that declares it.
- Built-in variables are not propagated between drivers. This includes any variables you define to override a built-in variable. The built-in variables for each driver are handled separately.
- Other prompting is handled unchanged at the beginning of each driver configuration file's import sequence.
- Refer to "Flexible Prompting in a Driver Configuration File" on page 171 for information about prompt layering supported by flexible prompting.

Flexible Prompting in a Driver Configuration File

`variable-decl` blocks can be marked to allow them to be prompted for separately, based on user input.

```
DTD changes:
-----------
* <!ENTITY % CompareMode "equals | not-equals">

  <!--******************************************************* -->
  <!--The variable-decl element contains definitions of variables   -->
  <!-- whose values can be prompted for and referred to throughout  -->
  <!-- the pre-configured driver file.                              -->
  <!-- ********************************************************** -->
  <!ELEMENT variable-decl(
    node-var*,
    text-var*)>
*  <!ATTLIST variable-decl
*    <!-- The following are used in the support of flexible -->
*    <!-- prompting.                                         -->
*    use-when-var  CDATA #IMPLIED
*    use-when-value  CDATA #IMPLIED
*    use-when-mode  (%CompareMode) "equals"
  >

* Added for flexible prompting.
```

Semantics

1. All `variable-decl` blocks with no `use-when-var` attribute are added to the prompt set.
2. All `variable-decl` blocks with a `use-when-var` attribute where the variable is defined and the variable value meets the condition are added to the prompt set.

   Variable analysis includes built-ins and variables carried forward from any previous import.
3. The user is prompted.
4. The prompt set is emptied and Steps 2 and 3 are repeated until there are no more prompts to process or all `variable-decl` blocks have been processed.
5. The import proceeds as before.

NOTE: The comparisons for `use-when-var` variables are case-insensitive.

Example 1

```
     <variable-decl use-when-var="varCheck" use-when-value="Fu" use-when-
mode="equals">
    <text-var prompt="When Fu?" var-name="fuVar"/>
  </variable-decl>

  <variable-decl use-when-var="varCheck" use-when-value="Fu" use-when-
mode="not-equals">
    <text-var prompt="When not Fu?" var-name="fuVar"/>
  </variable-decl>

  <variable-decl>
    <text-var prompt="Which other <variable-decl>?" var-name="varCheck">
      <dropdown>
        <value>Fu</value>
        <value>Bar</value>
      </dropdown>
    </text-var>
  </variable-decl>
```

In this example, the user would be prompted with a drop-down list. The description of the drop-down list is “Which other <variable-decl>?” The options in the list are Fu and Bar.

If the user select Fu from the drop-down and clicks **Next**, he or she is prompted again with a box. The description of the box is “When Fu?”

If the user selects anything else from the drop-down list and clicks **Next**, he or she is prompted with another box. The description of the box is “When not Fu?”

Example 2

```
<variable-decl use-when-var="varCheck" use-when-value="Fu">
    <text-var prompt="When Fu?" var-name="fuBarVar"/>
  </variable-decl>

  <variable-decl use-when-var="varCheck" use-when-value="Bar">
    <text-var prompt="When when Bar?" var-name="fuBarVar"/>
  </variable-decl>

  <variable-decl>
    <text-var prompt="Which other <variable-decl>?" var-name="varCheck"/>
  </variable-decl>
```

In this example, the user is presented with a box. The description of the box is “Which other <variable-decl>?” If the user specifies “Fu” in the box and clicks **Next**, he or she is presented with another box. The description on the second box is “When Fu?”

If the user specifies “Bar” in the box and clicks **Next**, he or she is presented with a box. The description is “When Bar?” If he or she specifies anything else, there are no further prompts and the variable fuBarVar is not defined.

Viewing the Informal Identity Manager Driver Configuration DTD

To view the informal Identity Manager Driver Configuration DTD, go to PCDrivers.txt (../samples/PCDrivers.txt). The DTD cannot be used for validation. It is not a valid XML DTD. It is a mechanism to document the valid constructs in a driver configuration file.

23 When and How to Use Global Configuration Values

Global Configuration Values (GCVs) are settings that are similar to driver parameters. GCVs are constant values, not global variables. There is no way to change a GCV value at runtime. The GCVs are globally accessible to the driver and driver set, but not to the tree or network.

Given these facts, GCVs are constants and cannot be changed at runtime, but they can be consumed by all drivers in a driver set or by all policies in a driver. This makes GCVs a very powerful configuration tool. Use the guidelines in the following sections when developing driver configuration files with GCVs.

Using GCVs to Adapt the Driver Configuration File to Changing Environments

GCVs help driver configuration files adapt to changing environments by externalizing environment-specific, such as placement contexts, domain names, IP addresses, usernames, dates, and times.

It is a bad practice to configure the driver to prompt for information during import and then add the answer directly into policy code. The better approach is to store the answer in a GCV and make the policies reference the GCV. If the answer to the prompt is wrong or the environment changes, the answer also needs to change. It is much simpler to change a single GCV value than to go through all policies that have the value and change the value in each policy.

By adding the configuration data as a GCV, the GCVs become the controls of the driver.

When Not to Use GCVs

If there are certain configuration values that must never change, they should not be surfaced in a GCV.

All information in a GCV can be easily changed or tuned. To keep certain configuration values and implementation details from being changed, add this information directly into the code. Everything that an administrator should see and be able to change should go into a GCV. Everything that only a developer sees or changes should go directly into the driver policy code.

The only reason to add a static value to a GCV is if it is used in many different places and it does not make sense to add it directly to the code.

When to Use Driver Set GCVs, Driver GCVs, or Global Configuration Objects in Packages

GCVs can be defined on a driver or driver set. The GCV location determines its scope and visibility. GCVs defined on the driver set are visible to all drivers and their policies in that driver set. GCVs defined on a driver can be consumed only by policies of the driver.

You can also create Global Configuration objects that contain GVCs and should be used when the configuration values are being referenced from content contained in packages. GCVs on the driver or driver set cannot be packaged. However, GCVs within a Global Configuration object can be installed, upgraded, downgraded, or removed with packages.

The scope of the Global Configuration object is determined by which driver or driver set the Global Configuration object is included in the Global Configuration list. Each driver and driver set has a Global Configuration list. The list is edited through the driver or driver set properties. On a driver, the Global Configuration list is located under the Driver Configuration option. On a driver set, the Global Configuration list is located under the Configuration option.

The GCV definitions in a Global Configuration object are identical to the GCVs that are contained on a driver or driver set. The precedence of GCV is as follows:

- GCVs from the driver
- GCVs from the driver Global Configuration list (Global Configuration objects)
- GCVs from the driver set
- GCVs from the driver set Global Configuration list (Global Configuration objects)

If a GCV defined on a driver has the same name as a GCV defined on the driver set, the driver GCV takes precedence in all policies that belong to that driver. All drivers that do not explicitly define the GCV on the driver, inherit this GCV from the driver set.

Global Configuration objects can be contained in a driver, driver set, or library. For more information, see "Global Configuration Value Definition Editor" in the *NetIQ Identity Manager - Using Designer to Create Policies* guide.

Naming Convention for GCVs

The GCV naming convention addresses the different types of GCVs, the different purposes of the GCVs, and the scopes of the GCVs.

```
[<purpose/scope>.]<group>.[<subgroup>.]<name>
```

- **Purpose/Scope:** The prefix idv (Identity Value) is used with the driver set GCVs. Driver-specific values are drv or driver.
- **Group:** Groups GCVs that belong together. Examples for groups could be communications, notification, logging, or security.
- **Subgroup:** Form subgroups within groups, such as smtp or snmp.
- **Name:** Descriptive name for the GCV.

For example:

```
idv.notification.smtp.ip
idv.notification.smtp.user
idv.notification.smtp.pwd
idv.notification.snmp.ip
idv.dit.data.locations
idv.dit.system.rbs
driver.samba.server
```

NOTE: Custom driver GCV values with special characters must be encoded.

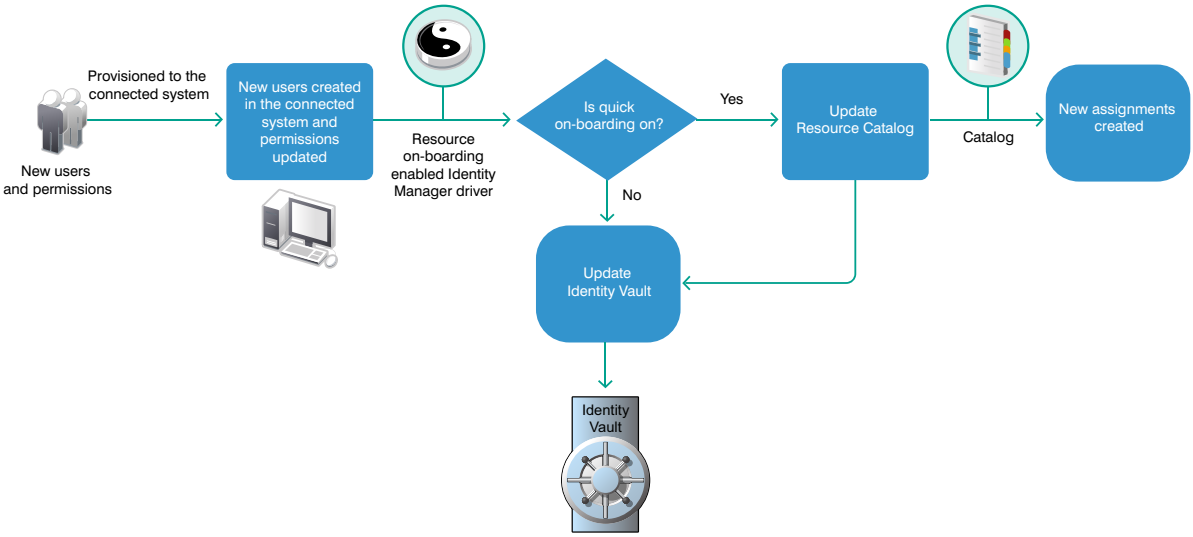
For example, `window%E2close()`

---

24 Extending Custom Entitlements

A driver entitlement represents a permission in an application (for example, Active directory) such as group membership, account, or any other entitlement or permission that an application uses. When a user is granted an entitlement in Identity Manager, a user attribute `DirXML-EntitlementRef` is generated with reference to the Identity manager driver entitlement and an entitlement value. A driver having this entitlement performs provisioning based on the type of the entitlement (group, account, role, and so on).

You can dynamically create resources with custom entitlements with permission values from a connected system, and also create permission assignments between Identity Manager resource model and connected systems. The following figure depicts how permissions flow from a connected system to the Resource Catalog and then into the Identity Vault.



Understanding the Resource Model

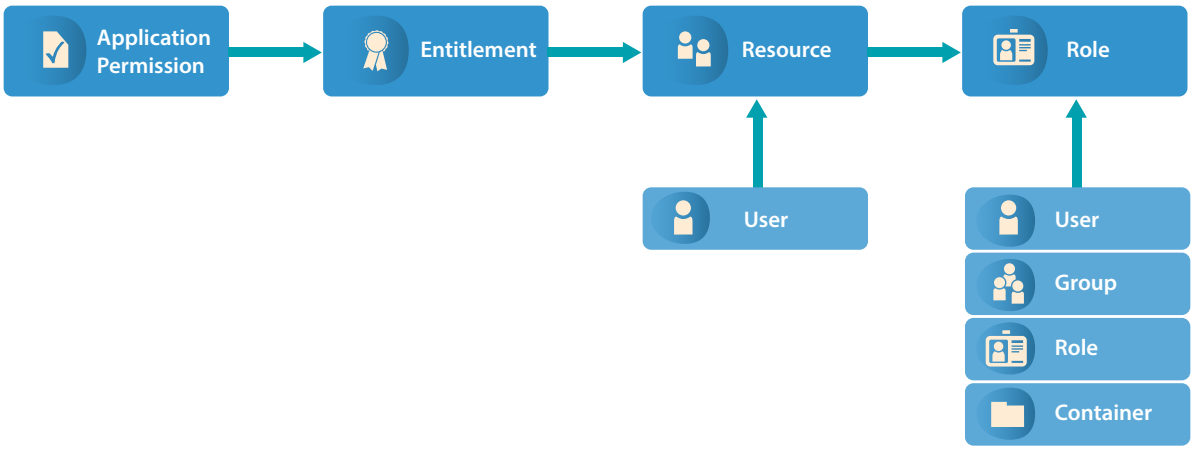
In Identity Manager, a resource is any digital entity such as a user account, computer, or database that a business user needs to be able to access.

Resources are similar to entitlements and used in user provisioning. Technically, you can think of a resource as an additional abstraction layer between driver entitlements and roles. Identity Manager restricts that a resource be associated with only one entitlement. However, you can bind a resource to the same entitlement more than once, with different entitlement values for each resource.

Resources can be assigned to roles. There are three levels of roles provided in the roles-base provisioning model. A role can be made up of other roles, which in turn can be made of other roles. At the lowest level a role has a resources attached to it.

You can map resource assignments to users or to roles within your organization. For example, you can use resources to:

- Make resource requests for users
- Create resources and map them to entitlements



Identity Manager leverages its resource model for performing event-based reconciliation of external system permission assignments. The resource model simplifies the entitlement model and provides you a convenient way to perform resource-based provisioning actions. The resource-based provisioning actions allow you to manage resource definitions and resource assignments within your organization.

You can assign resources only to users. You cannot assign them to groups or containers. If a resource is mapped to a role, that role can be assigned to a group or container resulting in an assignment of mapped resource to all the users in that group or container.

Before you can assign resources to users, the resources must be defined in Identity Manager. Identity Manager stores resources in the Resource Catalog of the User Application. The Resource Catalog also stores the supporting data needed by the Role and Resource Subsystem, which is the underlying infrastructure for roles-based provisioning module. All the defined resources are displayed in the Resource Catalog. You can create new resources, and modify, delete, and assign the existing resources.

Identity Manager provides User Application for end users to request the resources they need. It also provides additional tools that administrators can use to define and manage resources such as Designer and resource administration capabilities of Identity Applications.

Configuring Custom Entitlements

You use Designer to configure custom entitlements on a driver. The entitlements packages contain the content necessary for collecting and reconciling permissions. If you want a driver to support permission collection and reconciliation, ensure that these packages are installed on the driver. You can turn this functionality on or off using the new set of GCVs included with the driver.

You can either create a new driver with the latest packages or upgrade the packages for an existing driver. In both cases, you install the driver packages and then modify the driver configuration to suit your environment. For creating new drivers, NetIQ recommends that you refer to the individual driver documentation guides (https://www.netiq.com/documentation/identity-manager-49-drivers/).

Deploying Custom Entitlement Package for Identity Applications

Perform the following tasks to make custom entitlement available for Identity Applications using Designer:

1. Install custom entitlement package. This package automatically installs the CPRS common package.
2. Create an entitlement (`DirXML-Entitlement`) object for the driver. For example: `Cubicle`

   For more information, see Creating Entitlements through the Entitlement Wizard in the*NetIQ Designer for Identity Manager Administration Guide*.
3. To make the new entitlement available for Identity Applications Resources perform the following tasks:

   NOTE: To access Identity Applications Resource Catalog, navigate to **Administration > Resources** in Identity Manager Dashboard. For more information, see Listing Resources in *NetIQ Identity Manager - Administrator's Guide to the Identity Applications*.

   3a Right-click the driver and click **Properties**.

   3b Navigate to **GCVs > Custom Entitlements** tab.

   3c In **List of Custom Entitlements**, add the entitlement object names that needs to be listed in the Identity Applications user interface.

   NOTE: ◆Entitlement names are case sensitive.

   - CPRS supports only the Identity Manager 4.0 and later entitlement formats.For more information, see Entitlements Formats in the *NetIQ Identity Manager Entitlements Guide*.
   - By default, `cprs-supported`, `role-mapping`, and `resource-mapping` flags are set to **True** and the `data-collection` flag is set to **False**.

   3d (Conditional) To configure flags, create a GCV of boolean type in the following format:

   `drv.cprssupported.<entitlement-name>`, `drv.datacollection.<entitlement-name>`, `drv.rolemapping.<entitlement-name>`, `drv.resourcemapping.<entitlement-name>` respectively.

3e To include an additional XML to an entitlement:

3e1 Specify the **Name** as `drv.entitlement.extensions.<entitlement-name>`.

3e2 Select the **Type** as **String**.

3e3 Select the **Multi-line** option.

You should add the additional entitlement extensions between `<entitlement-extensions></entitlement-extensions>` XML node. For more information, see "Modifying Custom Entitlement Extension" on page 182.

3f (Conditional) For localizing the entitlement, add the localization values for the entitlement in the `L10N_<locale>` mapping table.

4 Deploy and start the driver for the changes to take effect.

Modifying Custom Entitlement Extension

Identity Manager supports the following XML tags for use with the entitlement node in the entitlement object XML. You must include these tags in the entitlement XML.

Parameters: If the entitlement is of type Identity Manager 4.0 or later, then User Application uses information from the parameters node to build the code map refresh values. You must define each key in the code map value in the `parameter` tag. For more information, see DTD.

account: For an account entitlement, you must define the following items under the `<account>` node:

- `<account-id>`: Contains details of the account unique ID in the User Application.
- `<account-status>`: Contains information about the attribute that stores the login or logout status of the user.

For more information, see DTD.

member-assignment-query: If the query for obtaining permission assignments is different than the query defined in the DirXML-entitlement object, you must redefine the query under this node. For more information, see DTD.

member-assignment-extensions: In case of multi-valued entitlements, if permission details are obtained from a specific attribute of the connected application, you must provide details of that attribute in this node. For more information, see DTD.

The following is a sample of entitlement extension XML for `<parameters>` and `<member-assignment-extensions>` tags:

Example 1:

To manage custom entitlement in the connected application such as `Cubicle`, where the permission value is stored in the `AssignedTo` attribute.

```
<entitlement-extensions>
  <parameters>
      <parameter mandatory="true" name="ID" source="association" />
            <parameter mandatory="true" name="ID2" source="association" />
  </parameters>
      <native-value source="src-dn" />
  <member-assignment-extensions>
            <query-xml>
        <read-attr attr-name="AssignedTo" />
            </query-xml>
  </member-assignment-extensions>
  </entitlement-extensions>
```

Example 2:

In this example, `<account>` and `<member-assignment-query>` nodes are defined for a multi-valued entitlement.

```
<entitlement-extensions>
      <account>
      <account-id source="src-dn" />
            <account-status active="FALSE" inactive="TRUE" source="read-attr" source-
name="Account_STATUS" />
      </account>
  <parameters>
          <parameter mandatory="true" name="ID" source="read-attr" source-
name="RESTACCOUNT" />
      </parameters>
  <member-assignment-query>
          <query-xml>
        <nds dtdversion="2.0">
                    <input>
             <query class-name="User" scope="subtree">
                              <search-class class-name="USer" />
             </query>
                    </input>
        </nds>
            </query-xml>
  </member-assignment-query>
  </entitlement-extensions>
```

25 Synchronizing Permission Changes from the Connected Systems

Permission Reconciliation Services (PCRS) is simplified to Controlled Permission Reconciliation Services (CPRS). CPRS currently supports only Active Directory, Multi-Domain Active Directory (MDAD) and LDAP drivers. PCRS is supported for all other drivers.For more information about CPRS, see Using Controlled Permission Reconciliation Services in the *NetIQ Identity Manager - Administrator's Guide to the Identity Applications*.

You need additional functionality to synchronize the permission assignment changes from the connected systems to Identity Manager. A permission assignment changes when a connected system administrator provides additional permissions to the existing users or creates new users. In this case, an Identity Manager driver publishes these changes to the Identity Vault, but the changes are not directly reflected in the User Application Resource Catalog because the default content shipped with the driver does not have this capability. To reflect the current state of a connected system in the Resource Catalog, you need to customize the package content of the driver.

Identity Manager provides Permission Collection and Reconciliation Service (PCRS) that enables you to create custom entitlements for connected system roles and resources in order to synchronize the connected system permission assignment changes to the User Application Resource Catalog. When PCRS is enabled, you can update a resource when permission assignments are published to the Identity Vault as they occur in connected systems.

PCRS provides the following key features:

- Supports easy creation of entitlements
- Provides out-of-box support for implementing Identity Manager resource model
- Supports onboarding of application permissions and assignments
- Supports assignment status updates on Publisher and Subscriber channels
- Supports bidirectional flow of resources and entitlements
- Reconciles resource or permission assignments between the Identity Vault and connected systems
- Provides integration between applications
- Supports comprehensive permission catalog with actual status display
- Provides a common package for custom drivers

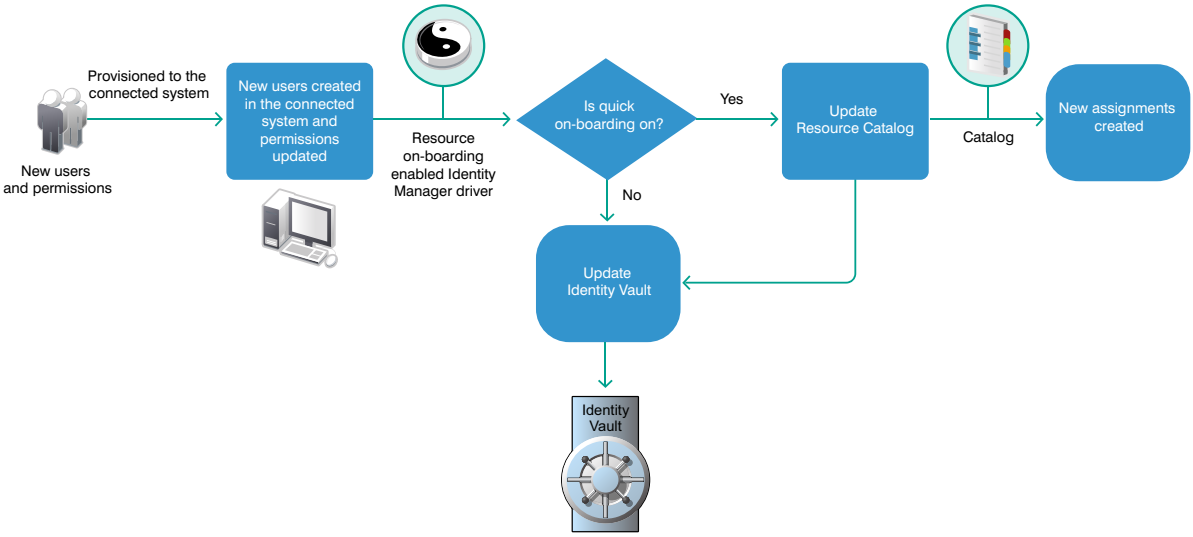
An Identity Manager driver installed with the package content for enabling PCRS can update the Resource Catalog with the connected system changes. The driver can automatically assign or revoke resources to Identity Manager identities based on the changes made to the attribute values in the connected system.

For a newly installed driver with this package content, you can migrate users and groups (for example, Active Directory) into the Identity Vault, which updates the Resource Catalog with the current state of the connected system. For example, if `User1` and `User2` are part of `Group1` with the required permissions in a connected system, a driver enabled with PCRS updates the user

permissions in the RBPM when the users are migrated from the connected system to the Identity Vault. The PCRS policies receive this event (migration) and update the Resource Catalog with the resource assignments.

NOTE: NetIQ recommends that you migrate individual users from a connected system to the Identity Vault instead of migrating groups. Migrating a group is not recommended because of performance issues.

You can dynamically create resources with custom entitlements with permission values from a connected system, and also create permission assignments between Identity Manager resource model and connected systems. The following figure depicts how permissions flow from a connected system to the Resource Catalog and then into the Identity Vault.



This section provides information about implementing PCRS in your Identity Manager environment.

- "Planning Overview" on page 186
- "Preparing Your Environment" on page 191
- "Viewing Permission Collection and Reconciliation Service Configuration Objects" on page 197
- "Troubleshooting Permission Collection and Reconciliation Service Issues" on page 198

Planning Overview

This section provides valuable information for planning a PCRS implementation in your Identity Manager environment.

Prerequisites

Ensure that you meet the following requirements for implementing PCRS:

- Designer for Identity Manager 4.0.2 AU4 and later
- Identity Manager 4.0.2 with Engine Patch 4 and later
- Managed System Gateway driver version 4.0.0.6 and later

- Driver Set Package:
  - Common Settings Advanced Edition Package (NOVLACOMSET 2.0.0 and later)
- Driver Package:
  - Driver-specific entitlements packages for Active Directory, LDAP, Delimited Text, and Loopback drivers

    For example, use NOVLADENTEX for Active Directory driver, NOVLLDAPENT for LDAP driver, and NOVLDTXTENT for Delimited Text driver. For more information, see driver-specific implementation guides.
  - NOVLCOMPCRS 2.0.0 and later PCRS package

    This is the common PCRS package for defining custom entitlements on drivers such as SOAP and JDBC.

Understanding the Resource Model

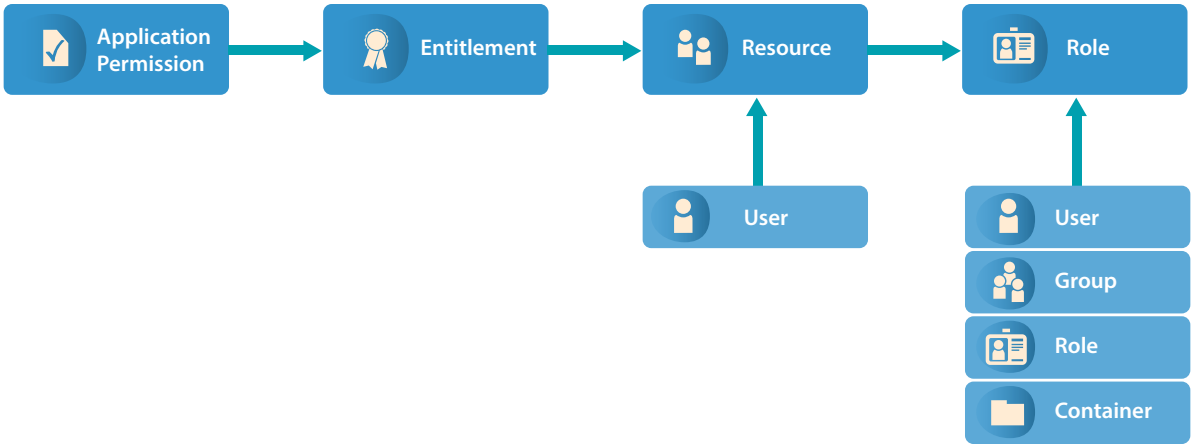
In Identity Manager, a resource is any digital entity such as a user account, computer, or database that a business user needs to be able to access.

Resources are similar to entitlements and used in user provisioning. Technically, you can think of a resource as an additional abstraction layer between driver entitlements and roles. Identity Manager restricts that a resource be associated with only one entitlement. However, you can bind a resource to the same entitlement more than once, with different entitlement values for each resource.

Resources can be assigned to roles. There are three levels of roles provided in the roles-base provisioning model. A role can be made up of other roles, which in turn can be made of other roles. At the lowest level a role has a resources attached to it.

You can map resource assignments to users or to roles within your organization. For example, you can use resources to:

- Make resource requests for users
- Create resources and map them to entitlements



Identity Manager leverages its resource model for performing event-based reconciliation of external system permission assignments. The resource model simplifies the entitlement model and provides you a convenient way to perform resource-based provisioning actions. The resource-based provisioning actions allow you to manage resource definitions and resource assignments within your organization.

You can assign resources only to users. You cannot assign them to groups or containers. If a resource is mapped to a role, that role can be assigned to a group or container resulting in an assignment of mapped resource to all the users in that group or container.

Before you can assign resources to users, the resources must be defined in Identity Manager. Identity Manager stores resources in the Resource Catalog of the User Application. The Resource Catalog also stores the supporting data needed by the Role and Resource Subsystem, which is the underlying infrastructure for roles-based provisioning module. All the defined resources are displayed in the Resource Catalog. You can create new resources, and modify, delete, and assign the existing resources.

Identity Manager provides User Application for end users to request the resources they need. It also provides additional tools that administrators can use to define and manage resources such as Designer and Catalog Administrator.

Understanding the Components for PCRS

The following Identity Manager components help you reconcile the connected system permissions with the Resource Catalog of the User Application.

- "Common Settings Advanced Edition Package GCVs" on page 188
- "Identity Manager Policy Objects" on page 189
- "PermissionOnboarding Job Object" on page 189
- "Mapping Table Objects" on page 190
- "Understanding the CSV File Format" on page 190

Common Settings Advanced Edition Package GCVs

The Common Settings Advanced Edition package includes the `NOVLACOMSET-GCVs` object, which contains two new GCVs used in permission assignment onboarding process.

- User Application Provisioning Services URL GCV (`UAProvURL`)
- User Application Provisioning Services Administrator GCV (`UAProvAdmin`)

The new entitlement packages create these GCVs which are used by the `PermissionOnboarding` job. For information about configuring the Common Settings GCVs, see "Configuring Common Settings GCVs" on page 192. For information about `PermissionOnboarding` job, see "PermissionOnboarding Job Object" on page 189.

Identity Manager Policy Objects

The Startup policies update the following mapping table objects:

- **PermissionNameToFile:** Contains entitlement configuration data that you specified in the Entitlements Name to CSV File Mappings page when the driver was created in Designer. You can add custom entitlements to this table.
- **PermissionEntMapping:** Created empty but is populated by the Startup policies and `PermissionOnboarding` job. It contains the configuration data transferred from the `PermissionNameToFile` object and DNs of entitlements created by the Startup policies. It also contains the LDAP DNs of the default dynamic User Application resource objects that are used to assign entitlements to users. You should not change the data populated by the Startup policies in this table.
- **StaticValueEntitlementMap:** Created empty but contains mapping between specific native entitlement values and a User Application static resource DN used by the driver to reconcile that value. You need to populate this table if you want to synchronize assignments bound to a static resource.

  You must manually enter the complete DN of the static resource in the `Static Resource` column.

IMPORTANT: Restart the driver for it to take effect of any changes made to the `PermissionNameToFile` and `StaticValueEntitlementMap` mapping table objects.

PermissionOnboarding Job Object

The driver policies update the `PermissionOnboarding` job parameters and the `PermissionEntMapping` mapping table.

The `PermissionOnboarding` job is a standard Identity Manager job and part of the entitlement package. The driver creates the job in the Identity Vault when the driver is deployed. The job runs when the driver starts. However, you can schedule the job to run periodically. Also, you can run it manually to process an updated CSV file. The NOVLCOMPCRS-ENT-startup-UpdateJobConfiguration Startup policies configure the `PermissionOnboarding` job.

The `PermissionOnboarding` job performs the following tasks:

- Reads the driver's `PermissionEntMapping` table object to obtain the list of the driver's entitlement objects populated by the Startup policies.
- Creates or verifies the existence of a dynamic `nrfResource` object to allow the assignment of the native permissions for each entitlement object in the `PermissionEntMapping` table. To do this, the job uses the Identity Manager provisioning, resource, and service SOAP APIs.
- Updates the `PermissionEntMapping` table with the nrfResource DNs.
- Reads the CSV file and populates the associated `<name>_Values` resource object with the values, display names, and descriptions for each entitlement object that specifies a CSV File catalog source in the `PermissionEntMapping` table.
- Calls a User Application private API to flush the User Application Entitlement cache so that newly created entitlements are recognized.
- Calls the User Application Entitlement refresh API to force the User Application to issue an entitlement query to obtain the catalog values for each driver entitlement.

Mapping Table Objects

When you install the entitlement package, the policies of this package are added to the driver Startup policy set. The driver executes these policies only once when the driver is started. The driver policies automatically configure the following objects for your environment:

- **Entitlement Objects:** The driver creates new entitlement objects to represent the native entitlement names identified in the `PermissionNameToFile` mapping table. The name of entitlement is the value of the `entitlementName` column of the `PermissionNameToFile` mapping table. For more information about mapping table objects, see Mapping Table Objects.
- **Permission Value Resource Object:** The driver adds a new permission value resource object, `entitlementName_values`. This object contains values for `Entitlement Values`, `Description`, and `Display Name` for every custom entitlement from the CSV file. It might also contain mappings for legacy value formats. To add more values to the entitlements, update the CSV file.
- **Entitlement Configuration Resource Object:** The driver creates a new entitlement configuration resource object, `EntitlementConfiguration`.

Understanding the CSV File Format

An Identity Manager driver enabled with PCRS consumes the custom entitlement information from a CSV file created by the system administrator of a connected system. The system administrator maintains a separate CSV file for every custom entitlement. A CSV file must contain values of the connected system permissions in the below format that Identity Manager can consume.

Attribute	Description
entitlement value	The custom entitlement value you want to add to the Resource Catalog.
entitlement display name	The custom entitlement name you want to display in the User Application.
entitlement description	The description for the new custom entitlement value.

The values for `entitlement display name` and `entitlement description` are consumed by the User Application while assigning a resource to a user. These attributes can be viewed in the User Application. For example, a CSV file containing details about granting access to the employees for a BuildingAccess entitlement represents this information in the following format:

```
Building A, Engineering, The engineering building
Building B, Accounting, The accounting building
Building C, Facilities, The facilities building
Building D, Warehouse, The warehouse
```

where *Building A* is the entitlement value, *Engineering* is the display name in the User Application for the entitlement value *Building A*, and *The engineering building* is the description for the entitlement value that appears in the User Application.

It is mandatory that the CSV file is placed on the Identity Manager server.

Preparing Your Environment

Your environment for PCRS must be configured appropriately. For example, the User Application must have new administrative user accounts that can be used by the policies to create and configure the configuration objects and job for quick onboarding of identity, resources, and permission assignments. This section helps you prepare your environment before you install a driver enabled with PCRS.

Setting Up Administrative User Accounts

To create and configure the configuration objects and job for quick onboarding of identity, resources, and permission assignments, the new policies use two administrative accounts: the **Administrative user** for the Identity Vault and a **Resource Administrator** for the User Application:

- **Identity Manager Driver Administrator:** Set this administrative user in the driver's Security Equivalence attribute when you deploy the driver. The policies use the user specified by the Security Equivalence attribute. This user needs the rights to create and modify the driver policy objects and to execute the `PermissionOnboarding` job, which is part of the driver package.
- **User Application Resource Administrator:** This administrative user performs the following tasks:
  - Creates resource objects
  - Triggers cache flush and entitlement refresh actions
  - Assigns and revokes resource assignments

For the User Application Resource Administrator user, NetIQ recommends that you create a new user. For example, you can create the new user, `cn=ResourceAdmin,OU=sa,O=data` and configure the following rights for the user in the User Application:

- **Role, Resource Creation, and Assignment Rights:** Click **Administration > RBPM Provisioning and Security > Administrator Assignments > Assign System Role Administrator**, select **Domain** and assign the following rights to the user:
  - Provisioning: Select All Permissions
  - Resource: Select All Permissions
  - Role: Select All Permissions

  NOTE: You need to assign each set of Domain permissions separately for the user.

- **Application Cache Refresh Rights:** In the User Application, click **Administration > Application Configuration > User Application Administrator Assignment**, then add the user to the **Current Assignments** list.

  NOTE: This user account is also used to filter the duplicate entitlement events occurring on the Subscriber channel as a result of auto-reconciliation of resource assignments.

Setting Up Administrative Passwords

To simplify the configuration and security of Identity Manager drivers with PCRS, use Distribution Password for the Driver Administrator user and the User Application Resource Administrator administrative user. Identity Manager uses Distribution Password to control password synchronization between the Identity Vault and connected systems. Identity Manager reads passwords for these administrative users from their `nspmDistributionPassword` attribute.

When you create the new administrator accounts, you must assign a password policy to the new user accounts. Optionally, you can use the default Sample Password Policy and assign the policy to them. For more information, see the *NetIQ Identity Manager - Administrator's Guide to the Identity Applications*.

Configuring Common Settings GCVs

The Common Settings Advanced Edition package contains the User Application Provisioning Administrator and User Application Provisioning URL GCVs. These GCVs are created by the new package supporting PCRS and used by the `PermissionOnboarding` job.

You must configure the GCVs on the driver set containing the drivers for quick onboarding of custom entitlements. Identity Manager recommends that you install and configure the GCVs before creating or configuring a new driver.

1. Start Designer and open your project.
2. In the Outline view, right-click **Package Catalog** and select **Import Package**.
3. Clear **Show Base Packages Only**.
4. Select the **Common Settings Advanced Edition** package and click **OK**, then click **OK** when Designer finishes importing the package.
5. In Modeler, right-click the driver set where you want to create your new drivers and select **Properties**.
6. Click **Packages**.
7. Click the **Add package** icon, select **Common Settings Advanced Edition**, then click **OK**.
8. Click **OK**.
9. (Conditional) Click **OK** to install any additional package dependencies.
10. In the Package Installation Wizard, specify the URL for your User Application installation and the DN of your User Application Administrator account, then click **Next**.
11. Click **Finish**.

Now you have all the information for creating custom entitlements. For creating custom entitlements, continue with "Understanding the PCRS Process" on page 194.

Configuring Custom Entitlements

You use Designer to configure custom entitlements on a driver enabled with PCRS. The entitlements packages contain the content necessary for collecting and reconciling permissions. If you want a driver to support permission collection and reconciliation, ensure that these packages are installed on the driver. You can turn this functionality on or off using the new set of GCVs included with the driver.

You can either create a new driver with the latest packages or upgrade the packages for an existing driver. In both cases, you install the driver packages and then modify the driver configuration to suit your environment. For creating new drivers, NetIQ recommends that you refer to the individual driver documentation guides.

Installing the Driver Packages

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

1. In Designer, open your project.
2. In the Modeler, right-click the driver set where you want to create the driver, then select **New > Driver**.
3. Follow the driver configuration wizard to create the driver.
4. On the Entitlements Name to CSV File Mappings page, click the **Add Name to File Mapping** icon to populate the page with the entitlement configuration options.

   Identity Manager uses the CSV file to map entitlements to corresponding resources in the Identity Manager catalog.

   The information that you specify on this page is used for creating the permission catalog. This page uses *BuildingAccess* as an example. Fill in the following fields, then click **Next**:

   - **Entitlement Name:** Specify a descriptive name for the entitlement to map it to the CSV file that contains the connected system entitlement details.

     Entitlement Name is the name of the entitlement. This parameter corresponds to the Entitlement Assignment Attribute in the connected system. For example, you could define an entitlement called *BuildingAccess*.

     This parameter is used to create a resource in the User Application.

   - **Entitlement Assignment Attribute:** Specify a descriptive name for the assignment attribute for an entitlement.

     This parameter holds the entitlement values from the connected system. For the engine to receive the attribute events from the Publisher channel, ensure that you add this parameter to the Driver Filter. For example, you could have an attribute called *BuildingAccess*.

     NOTE: For the Delimited Text Driver, add this parameter in the **Field Names** on the Driver Parameters page or modify it in driver settings after creating the driver.

- **CSV File:** Specify the location of the CSV file. This file must be located on the same server as the Identity Manager engine. This file contains the values for the application entitlements in the format *value*, *displayname*, and *description*. An example for the CSV file path is `/root/user/BuildingAccess.csv`. For more information, see "Understanding the CSV File Format" on page 190.
- **Multi-valued?:** Set the value of this parameter to **True** if you want to assign resources and entitlements multiple times with different values to the same user. Otherwise, set it to **False**.

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

The driver is now created. Deploy the driver to the Identity Vault.

Understanding the PCRS Process

After a driver with custom entitlements is created or configured in Designer, you must deploy it into the Identity Vault. When the driver is started, the driver automatically creates new resources based on the entitlements configured and populates the resource with the entitlement values from the CSV file. The new custom entitlement and the corresponding resource object is updated in the **PermissionEntMapping** table. When a permission assignment changes in the connected system, the driver policies consume the modified permission value and update the Resource Catalog. The following sections provide information about the sequence of actions involved in this process.

Sequence of Actions

1 When the driver is started, the following actions occur:

1a Identity Manager executes the driver start up policies.

1b The `InitEntitlementConfigurationResource` policies perform the following actions:

- Reads the `permissionnametofile` mapping table to retrieve the entitlement information from this mapping table and creates these entitlements in the Identity Vault if they are not already existing.
- Updates the `PermissionEntMapping` object with the entitlement information and the location of the CSV file.
- Updates the entitlement configuration resource object with the entitlement information. RBPM uses this object for refreshing the code map.

1c The `SetJobNamedPassword` and `UpdateJobConfiguration` policies configure the permission onboarding job with the required permissions.

1d The `OnboardEntitlements` policy starts the `PermissionOnboarding` job.

When the job is started, it performs the following actions:

- Reads the CSV file containing the entitlement values.
- Populates the `<entitlementname>_Values` objects in the Identity Vault. For example, for a custom entitlement named `BuildingAccess`, it creates a DirXML-Resource object in eDirectory with name `BuildingAccess_values` containing the values of the custom entitlement.
- Creates a dynamic resource for assigning new custom entitlement values to the users.

- Populates the `PermissionEntMapping` object with the newly created resource name.
- Issues a code map refresh to update the Resource Catalog with the new entitlement values.

At the end of this step, verify that the following tasks are completed:

1. Custom entitlements are created in the Identity Vault.
2. The `EntitlementConfiguration` resource object is updated in the Identity Vault.
3. `<entitlementname>_Values` object is updated for each entitlement in the Identity Vault.
4. Resources are created in RBPM.
5. The `PermissionEntMapping` object is updated in the Identity Vault.

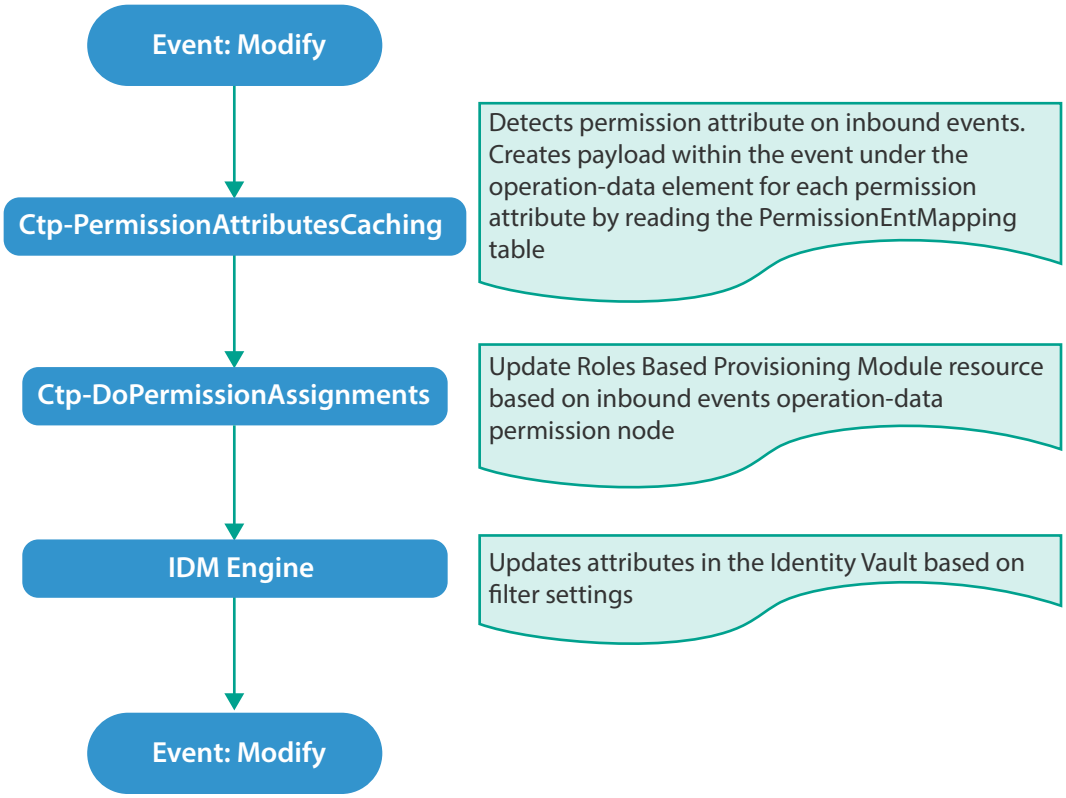
   Now the driver is ready to synchronize the connected system permission changes to the User Application.

For more information, see "Viewing Permission Collection and Reconciliation Service Configuration Objects" on page 197.

How Permissions Are Reconciled

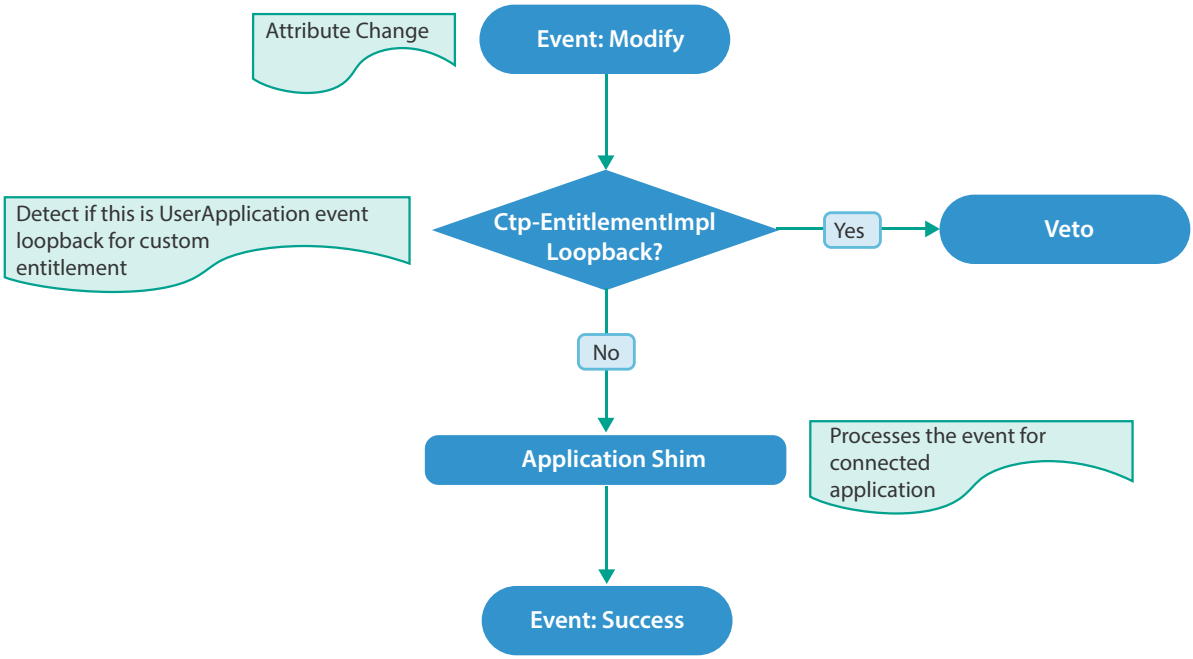
The below examples will help you understand how permissions are reconciled in different scenarios.

Use Case 1: When the Publisher channel has an event with permission changes.



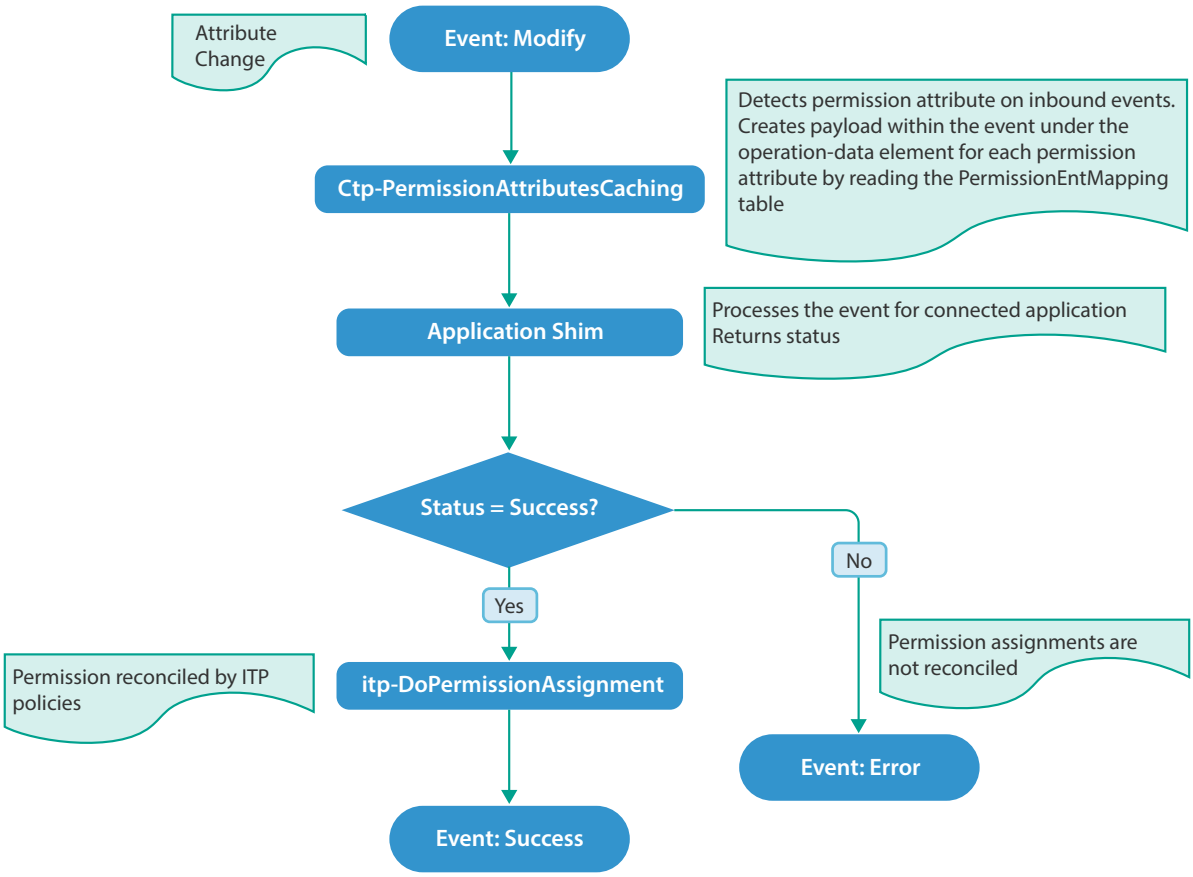
Use Case 2: When a permission attribute changes in the Identity Vault.

In this case, Identity Manager sends a Subscriber event to update the changed permission in the connected system. Also, the permission is reconciled in Resource Catalog.



Use Case 3: When a permission changes in Resource Catalog.

This action causes a Subscriber event to update the permission in the connected system.



IMPORTANT: You can create a new resource and map it to a custom entitlement and reconcile permission assignments. However, if PCRS is turned off, all resource assignments for the custom entitlement are disabled regardless of whether the resource is created through PCRS or not. Identity Manager does not support using the common package included with PCRS on a driver that already supports entitlements

Viewing Permission Collection and Reconciliation Service Configuration Objects

After the driver is deployed and configured with PCRS, verify that the driver correctly creates and updates the entitlements information in the Identity Vault.

Complete the following steps:

1 Open the Identity Manager Administration page.

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

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

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

3 Click the driver icon.

4 Click the **Jobs** tab. The `PermissionOnboarding` job appears on the Jobs page. For more information, see PermissionOnboarding Job Object.

5 Click **Advanced > Mapping Tables**. The DNs of the Entitlement objects appear on the Mapping Table page based on the InitEntitlementResourceObjects policy and data from the configuration objects. For more information, see Mapping Table Objects.

6 Click **Driver Set > Edit Driver Set properties**.

7 Click **Global Config Values** to display the driver set GCV page.

This page contains two sets of GCVs that are consumed by the drivers in the driver set. Ensure that you configure them for the driver set containing the drivers for quick onboarding of identity, resources, and permission assignments.

- **NOVLCOMSET:** This GCV object contains the following:
  - **User Container:** Specifies the Identity Vault container where the users are added, if they do not already exist in the Identity Vault. This value is the default value for all drivers in the driver set.
  - **Group Container:** Specifies the Identity Vault container where the groups are added, if they do not already exist in the Identity Vault. This value is the default value for all drivers in the driver set.
- **Advanced Settings:** This GCV object contains the following:
  - **User Application Provisioning Services URL:** Specifies the User Application Identity Manager Provisioning URL.
  - **User Application Provisioning Services Administrator:** Specifies the DN of the provisioning services administrator. This user should have the rights for creating and assigning resources.

Troubleshooting Permission Collection and Reconciliation Service Issues

The following known limitations and workarounds can help you troubleshoot issues that you might encounter while using the Permission Collection and Reconciliation service:

The driver ignores permission assignments

Explanation: If you use the same user (User Application Administrator account) that the driver uses to communicate with the User Application, the driver ignores these changes and treats them as loopback events.

Action: 1. Use a different User Application administrator account for the driver.

2. Check the JBoss `server.log` file in the User Application to determine if User Application encountered any error when `PermissionOnboarding` job made SOAP calls to the `server.log` file.

3. Set the `PermissionOnboarding` job trace level to 5 to verify if the job ran successfully and was able to perform a codemap refresh, create a resource, and update the `PermissionEntMapping` table with the resource DN.

4. Set the driver trace level to 5 if you want to view policy processing sequence.

The Subscriber Channel ignores permission reconciliation

Explanation: For any Subscriber changes, permissions are reconciled only after an event is successfully processed. The driver might not reconcile permissions if it contains policies that ignore `operation-data` containing permissions when these policies create or transform the status document.

Action: Restore `operation-data`.

For example:

```
<xsl:template match="operation-data">
<operation-data>
<xsl:message>operation-data</xsl:message>
<!-- ignore this element but process all children -->
<xsl:apply-templates select="node()|@*"/>
</operation-data>
</xsl:template>
```

Deleting an entitlement value in a connected application is not reflected in the mapped resource

Explanation: This occurs when the `Optimize-Modify` value is set to `yes`.

Action: Set the filter attribute value to `Notify`.

Resources are not created in RBPM

Explanation: This occurs when the resource DN value is already populated in the `PermissionEntMapping` table.

Action: Delete the resource DN value and restart the driver. Otherwise, run the `PermissionOnboarding` job.

Changes to mapped attributes in the Identity Vault are not reflected as assignments in RBPM

Explanation: The `NOVLCOMPCRS-itp-DoPermissionAssignment` policy takes care of reconciling permissions when an attribute is successfully updated in connected application. When the status document passes through this policy, the policy acts upon `operation-data`. If you added a new transformation policy in the policy set, ensure that `operation-data` remains unchanged in the status document.

Action: Verify if the `NOVLCOMPCRS-itp-DoPermissionAssignment` policy is placed correctly in the policy hierarchy, so that changes made to the attributes in the Identity Vault are reconciled to assignment attributes in RBPM.

26 Troubleshooting

The following sections describe various ways to troubleshoot the issues that you may encounter while working with Identity Manager:

Identity Manager Treats SYN_TIME values as Signed Integers Instead of Unsigned Integers

Identity Manager stores timestamps in the Identity Vault. The timestamps use a Timestamp syntax that uses an epoch number in a 32-bit attribute to count seconds from year 1970. The attribute is 32-bits long and imposes a limitation on the time range that can be specified.

Identity Manager engine versions prior to 4.5.x and 4.6.2 treat the timestamp value as a signed integer. The range can store dates prior to 1970, allowing dates to be specified between 1903 to 2037.

Identity Manager engine versions 4.5.x, 4.6.0, 4.6.1, 4.7 and 4.7.1 treat timestamps as unsigned integer (like eDirectory does for LDAP). This allows you to specify the time between 1970 until 2106.

To store timestamps outside these time ranges, it is recommended to use the SYN_INTEGER64 syntax.

Attributes Are Removed After Synthetic Add or Optimization Operations

When a `modify` event is converted to a `synthetic add` event in the Subscriber channel, the Identity Manager engine discards any attributes whose modification timestamps are greater than the modification timestamp in the current `modify` event. The engine does not include such attributes in the constructed add event. This may occur if some of the attributes are synchronized from a different eDirectory replica server.

Using NetIQ Sentinel to Log Identity Manager Events

You can log Identity Manager events by using NetIQ Sentinel. Using this service in combination with the driver log level setting provides you with tracking control at a very granular level. For more information, see the *NetIQ Identity Manager - Configuring Auditing in Identity Manager*.

Troubleshooting Driver Processes

Viewing driver processes is necessary to analyze unexpected behavior. To view the driver processing events, use DSTrace. You should use it only during testing and troubleshooting the driver. Running DSTrace while the drivers are in production increases the utilization on the Identity Manager server and can cause events to process very slowly. For more information, see Chapter 21, "Viewing Identity Manager Processes," on page 165.

Driver Shim Errors

This section identifies errors that might occur in the core driver shim. Error messages that contain a numerical code can have various messages depending on the application or Web service.

307 Temporary Redirect

Source: The status log or DSTrace screen.

Explanation: The Subscriber channel attempted to send data to the application or Web service but received a 307 Temporary Redirect response.

Possible Cause: The Web service is not available.

Action: The Subscriber waits for a period of time (usually 30 seconds) and tries again.

Level: Retry

408 Request Timeout

Source: The status log or DSTrace screen.

Explanation: The Subscriber channel attempted to send data to the application or Web service but received a 408 Request Timeout response.

Possible Cause: The Web service or application is busy.

Action: The Subscriber waits for a period of time (usually 30 seconds) and tries again.

Level: Retry

503 Service Unavailable

Source: The status log or DSTrace screen.

Explanation: The Subscriber channel attempted to send data to the application or Web service but received a 503 Service Unavailable response.

Possible Cause: The Web service or application is down.

Action: The Subscriber waits for a period of time (usually 30 seconds) and tries again.

Level: Retry

504 Gateway Timeout

Source: The status log or DSTrace screen.

Explanation: The Subscriber channel attempted to send data to the application or Web service but received a 504 Gateway Timeout response.

Possible Cause: The gateway is down.

Action: The Subscriber waits for a period of time (usually 30 seconds) and tries again.

Level: Retry

200-299 Messages

Source: The HTTP server.

Explanation: The messages in the 200-299 range indicate success.

Action: No action required.

Level: Success

Other HTTP Errors Messages

Source: The status log or DSTrace screen.

Explanation: Other numerical error codes result in an error message containing that code and the message provided by the HTTP server. In most cases, the driver continues to run, and the command that caused the error isn't retried.

Possible Cause: There are multiple causes for the different errors.

Action: See RFC 2616 (http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html) for a list of all HTTP error codes and explanations.

Level: Error

Problem communicating with HTTP server. Make sure server is running and accepting requests.

Source: The status log or DSTrace screen.

Explanation: The Subscriber channel received an IOException while communicating or attempting to communicate with the HTTP server.

Possible Cause: The HTTP server is not running.

Possible Cause: The HTTP server is overloaded.

Possible Cause: There are firewall restrictions blocking access to the HTTP server.

Possible Cause: The URL provided in the Subscriber configuration is not correct.

Action: Start the HTTP server.

Action: Remove services, if the HTTP server is overloaded.

Action: Change the firewall restrictions to allow access to the HTTP server.

Level: Retry

The HTTP/SOAP driver does not return any application schema by default.

Source: The status log or DSTrace screen.

Explanation: The driver is not returning any application schema, but the driver continues to run.

Possible Cause: The Identity Manager engine calls the DriverShim.getSchema() method of the driver, and the driver is not using the SchemaReporter customization.

Action: A Java class needs to be written that implements the SchemaReporter interface, and the driver needs to be configured to load the class as a Java extension.

Level: Warning

Subscriber.execute() was called but the Subscriber was not configured correctly. The command was ignored.

Source: The status log or DSTrace screen.

Explanation: The Subscriber channel of the driver isn't initialized properly. The driver continues to run but displays this message each time an event is received by the Subscriber channel.

Possible Cause: An improperly formatted driver configuration.

Action: Configure the driver correctly.

Action: Clear the Subscriber's filter so it doesn't receive commands.

Level: Warning

pubHostPort must be in the form host:port

Source: The status log or DSTrace screen.

Explanation: The driver cannot communicate.

Possible Cause: An error occurred with the Publisher channel configuration.

Action: Review the Publisher channel parameters to verify that both a valid host and a valid port number are provided.

Level: Fatal

MalformedURLException

Source: The status log or the DSTrace screen.

Explanation: There is a problem with the format of the URL.

Possible Cause: The URL supplied in the Subscriber channel parameters isn't in a valid URL format.

Action: Change the URL to a valid format.

Level: Fatal

Multiple Exceptions

Source: The status log or the DSTrace screen.

Explanation: The HTTP listener fails to properly initialize.

Possible Cause: There are a variety of reasons for this error.

Action: Check your Publisher settings to make sure you have specified a port that is not already in use and that the other Publisher settings are correct.

Level: Fatal

HTTPS Hostname Wrong: Should Be ...

Source: The status log or the DSTrace screen.

Explanation: An SSL handshake failed on the Subscriber channel.

Possible Cause: The subject presented with the server certificate doesn't match the IP address or hostname given in the HTTPS URL.

Action: Use a DNS hostname rather than an IP address in the URL.

Level: Retry

Identity Manager Driver Errors

The following errors might occur with Identity Manager drivers:

641 Unable to start the driver

Source: DSTrace screen.

Explanation: You might receive this error message when you start an Identity Manager driver through Designer.

Possible Cause: There are several causes. The error means the Identity Manager engine cannot load properly when eDirectory loads.

Action: Perform the following troubleshooting steps on the Identity Manager engine to know what exactly the engine is doing when it is loaded, and all jvmloader related messages.

Identity Manager engine running on Windows:

NOTE: Identity Manager is installed in the directory where eDirectory's dlms are present, by default, `C:\Novell\NDS`.

---

1. Stop the eDirectory service.
2. Move the `dirxml.dlm` file from the directory where eDirectory is installed.
3. Start the eDirectory service.
4. After eDirectory is loaded, start `DSTRACE.dlm`.

5 Click **Edit > Option**.

6 Set the following flags:

DirXML

DirXML Drivers

Misc Other

Deselect all other options and click **OK**.

7 Click **File > New** and specify a filename for your trace file.

8 Move the `dirxml.dlm` file back to its original location.

9 Close/reopen the eDirectory services console.

10 Select `dirxml.dlm` and click `Start`.

The trace file contains the messages related to why the VRDIM module (IDM Engine) does not start.

Identity Manager engine running on Linux:

1 Stop `ndsd` by using the following command:

```
/etc/init.d/ndsd stop
```

2 Move the `libvrdim.*` files from their original directory to a different directory.

The files are located in the `/opt/novell/eDirectory/lib/nds-modules/` directory.

3 Start `ndsd` by using the following command:

```
/etc/init.d/ndsd start
```

4 Start `ndstrace`.

5 In the `ndstrace`, type the following:

```
set ndstrace=nodebug
set ndstrace=+time
set ndstrace=+tags
set ndstrace=+misc
set ndstrace=+dxml
set ndstrace=+dvrs
ndstrace file on
```

6 Let `ndstrace` run on the screen.

7 Move the `libvrdim.*` files back to their original location.

8 In the `ndstrace` screen, type the following:

```
load vrdim
```

9 After you see the errors, stop `ndstrace`.

The trace file contains the messages related to why the VRDIM module (IDM Engine) does not start.

You can specify a different location for the trace file. To see the contents of a trace file, click **Edit Properties > Misc > Trace File**.

After you have performed the specified troubleshooting steps, you might encounter the "783 Unable to start the driver" on page 207error.

Level: Error

783 Unable to start the driver

Source: DSTrace screen

Explanation: You might encounter this error message after performing the Action specified under "641 Unable to start the driver" on page 205.

Possible Cause: There are several possible causes for the error. The following is a list of few possible causes with suggestions to help fix and/or track them further:

Possible Cause: Corruption in the association between the driver set and the server.

Action: Remove the association between the driverset and the server, cycle ndsd, and add the association back again.

Possible Cause: Damage/corruption in the DirXML-ServerKeys attribute that exists in the local DIB's Pseudo-Server object.

Action: DSDUMP (done only by technical support) is needed to remove the attribute with pre-Identity Manager 3.6. Use the new command within the dxcmd utility for Identity Manager 3.6 and later.

Possible Cause: Using Dib Clone or dsbk will cause damage or corruption in the DirXML-ServerKeys attribute that exists in the local DIB's Pseudo-Server object.

Action: DSDUMP (done only by technical support) is needed to remove the attribute with pre-IDM 3.6. Use the new command within the dxcmd utility for Identity Manager 3.6 and later.

Possible Cause: Misconfiguration of the JVM heap sizes.

Action: This is shown in the **+misc** flag in `ndstrace`. All jvmloader messages exist under the **MISC** flag on Linux/Unix, and the **MISC OTHER** flag on dstrace.dlm (Windows). You cannot see the jvmloader messages on Netware. But you can always check the `SYS:/ETC/JAVA.CFG` file.

Possible Cause: Corruption/damage/insufficient rights on the libraries IDM requires in the box (check the install log, also do an rpm -V on the Identity Manager packages)

Action: Try the following options:

- Use `rpm -V` liberally against all Identity Manager-related libraries. The command line works well for that:

  ```
  rpm -qa | grep DXML | xargs rpm -V
  ```

- Add your IDM libraries to see the dependencies they have, and check the dependencies also. The `checkbin.sh` script can be of great help when checking dependencies. It is part of the `ntsutils` package that can be downloaded from the following location:

  https://www.netiq.com/communities/cool-solutions//supportconfig-linux (https://www.netiq.com/communities/cool-solutions//supportconfig-linux)

- Use `strace/ltrace` tool to track what is happening. Because `strace` works on scripts, it is the best option. `ltrace` can be run only against binary files. These tools provide a lot of information, and quite a bit of it requires basic knowledge of C programming language.
- Reinstall Identity Manager on top of itself. The process overwrites the libraries/rights.

Level: Error

Identity Manager Driver that is deleted without stopping, and re-created with the same port number fails to start

Source: DSTrace screen

Explanation: Delete a driver without stopping it, re-create the same driver with the same port number, and start the driver. The driver fails to start.

Possible Cause: IDM does not release the used port of a driver if the driver is deleted without stopping.

Action: Edit the driver to use another port. The driver starts.

Java Customization Errors

The following errors might occur in the customized Java extensions.

SchemaReporter init problem: extension-specific message

Source: The status log or DSTrace screen.

Explanation: The SchemaReporter Java customization had a problem initializing, and the driver shuts down.

Possible Cause: The Java extension is not initialized correctly.

Action: Verify that the Java extension is enabled in the driver.

Level: Fatal

Extension (custom code) init problem: extension-specific message

Source: The status log or DSTrace screen.

Explanation: One of the following Java extensions failed to initialize:

- SubscriberTransport
- PublisherTransport
- DocumentModifiers
- ByteArrayModifiers

Possible Cause: The Java extension is incorrect.

Action: Review the Java extension and verify it is enabled in the driver.

Level: Fatal

Various other errors

Source: The interfaces provided for Java extensions return error messages on the trace screen and sometimes to the Identity Manager engine.

Explanation: Sometimes it is difficult to distinguish errors of this type from other errors that originate in the core driver shim. If you get errors that are not listed in this section and you are using Java extensions, check with whomever provided you with the extensions for a list of error codes for that particular extension.

Level: Varies

Multiple Sync Events Occur When an Object is Moved in the Master Replica Server

When Identity Manager is not in the master replica server and if some object is moved in the master replica server, there might be multiple sync events generated on the drivers. Some of these sync events are generated before the object move is completed in the master replica server and hence these sync events are not reliable. If some modifications are applied to the moved object in the Identity Manager server, based on any such event, the modifications are ignored because the move is not yet completed in the master replica server.

If you encounter such a situation, check for the timestamp on the <*sync*> event. If the timestamp shows a value `0#0`, it is reliable. If it shows any other value, then it is not reliable.

The following is an example of an unreliable sync event, because it has a timestamp other than 0#0 (marked in bold).

```
<nds dtdversion="3.5" ndsversion="8.x">
<source>
<product version="3.5.13.20090903 ">DirXML</product>
<contact>Novell, Inc.</contact>
</source>
<input>
<sync class-name="User" event-id="sles10sp3#20130722144515#2#2" from-
move="true" qualified-src-dn="O=novell\CN=Auser1" src-dn="\SLES10SP3-
TREE\Novell\Auser1" timestamp="1374504315#0">
<association state="associated">{D6B73031-695D-074e-AAAD-D6B73031695D}</
association>
</sync>
</input>
</nds>
```

The following is an example of a reliable sync event, because it has a timestamp 0#0 (marked in bold).

```
<nds dtdversion="3.5" ndsversion="8.x">
<source>
<product version="3.5.13.20090903 ">DirXML</product>
<contact>Novell, Inc.</contact>
</source>
<input>
<sync cached-time="20130722144516.217Z" class-name="User" event-
id="sles10sp3#20130722144515#2#2" from-move="true" qualified-src-
dn="O=novell\CN=Auser1" src-dn="\SLES10SP3-TREE\Novell\Auser1"
timestamp="0#0">
<association state="associated">{D6B73031-695D-074e-AAAD-D6B73031695D}</
association>
</sync>
</input>
</nds>
```

Rule Engine Does Not Honor the Mode Specified in the Set or Add Destination Attribute Value

When a rule set has more than one action to add additional attributes, at the time of adding an event to the destination object, the rule engine considers only the mode of the first attribute value. It ignores the modes of the attributes in the rest of the actions and applies the mode that you set for the first action.

For example, if the mode specified for the first action to add the destination attribute value is `after the current operation` (so that attribute value is set in a modify event after the add event) and the rest of the actions to set destination attributes have the `add to current operation` mode, the rest of the actions also end up being part of the modify event and not the add event, as expected.

To work around this issue, move all of the actions to add destination attributes with mode `after the current operation` to the end of the rule set. Then the attributes with `add to current operation` will be part of an add event and the rest will be in a separate modify event.

Reassociating a Driver Set Object with a Server

A driver set object is associated with a server. If the association becomes invalid for some reason, it is indicated by one of the following:

- When upgrading eDirectory on your Identity Manager server, you get the error `UniqueSPIException error -783`.
- No server is listed in the **Servers** tab on the driver or driver set.
- A server is listed next to the driver in the Identity Manager Overview screen, but the name is garbled text.

To resolve this issue, you must disassociate the driver set object and the server, and then reassociate them:

1. Click the **IDM Administration** tab from the Identity Console home screen.
2. Click Actions icon on the specific driver set object that you want to reassociate with a server.

3 Click **Remove server**.

4 Click **Add server**.

NOTE: When you reassociate a driver set object with a server, all drivers are disabled, and all passwords are cleared.

A Data Synchronization Flow

Identity Manager uses four main components to synchronize data between the Identity Vault and the connected system (external application).

- The Identity Manager engine that provides the framework.
- The Identity Manager policies that control the mapping of attributes and classes and the matching and creation of entries.
- Event filters that control the direction of data synchronization.
- The Identity Manager driver shim that serves as an interface between the application and the Identity Manager engine.

This section contains a brief explanation of the Identity Manager processes applied to data that flows between Identity Manager and the connected system.

- "The Identity Vault" on page 213
- "The Shim" on page 217
- "Channels" on page 218
- "Events and Commands" on page 218
- "Schema Mapping Policy" on page 219
- "Event Transformation Rule" on page 219
- "Filter" on page 220
- "Add Processor" on page 221
- "Matching Rule" on page 222
- "Create Rule" on page 223
- "Placement Rule" on page 224
- "Command Transformation Rule" on page 225
- "Rules, Policies, and Style Sheets" on page 225

The Identity Vault

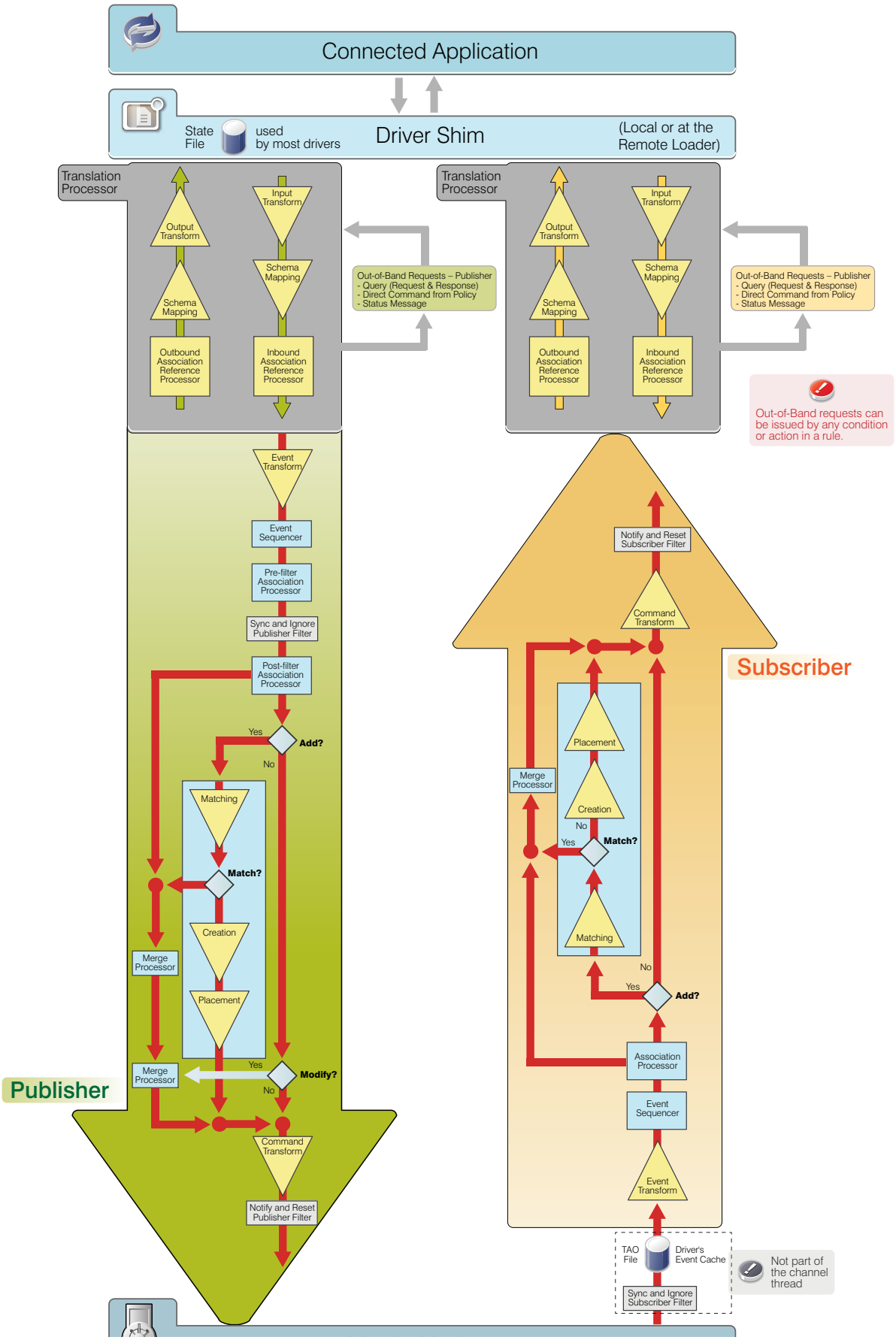
The Identity Vault is a repository of identity information. It is also called the NetIQ eDirectory tree. The Identity Vault stores information specific to Identity Manager, such as driver configurations, parameters, and policies.

The Identity Vault has an extensive schema which may be customized. The Identity Vault can be viewed narrowly as a private data store for Identity Manager or more broadly as a Identity Vault that holds enterprise-wide data, that you want to synchronize amongst applications including various directories, databases, phone systems, operating systems, and Human Resource systems. The data in the vault is available to any protocol supported by eDirectory, including NCP (NetWare Core Protocol), LDAP, and DSML. Identity Manager eases the administrative efforts of large enterprises by

preventing administrative effort duplication. For example, data synchronized from a Lotus Notes system to a SAP HR database is first added to the Identity Vault and then sent to the SAP HR database.

A typical Identity Manager environment has an Identity Vault at the center with other applications connected to it. The Identity Manager architecture can be thought of as multiple one-to-one relationships or a hub-and-spoke relationship. Each individual relationship is between the Identity Manager, Identity Vault, and a specific connected application.

Figure A-1 *Fishbone View*


Connected Application
State File
used by most drivers
Driver Shim
(Local or at the Remote Loader)
Translation Processor
Output Transform
Schema Mapping
Outbound Association Reference Processor
Input Transform
Schema Mapping
Inbound Association Reference Processor
Out-of-Band Requests – Publisher
- Query (Request & Response)
- Direct Command from Policy
- Status Message
Translation Processor
Output Transform
Schema Mapping
Outbound Association Reference Processor
Input Transform
Schema Mapping
Inbound Association Reference Processor
Out-of-Band Requests – Publisher
- Query (Request & Response)
- Direct Command from Policy
- Status Message
Out-of-Band requests can be issued by any condition or action in a rule.
Event Transform
Event Sequencer
Pre-filter Association Processor
Sync and Ignore Publisher Filter
Post-filter Association Processor
Yes
Add?
No
Matching
Match?
Creation
Placement
Merge Processor
Merge Processor
Yes
Modify?
No
Command Transform
Notify and Reset Publisher Filter
Publisher
Notify and Reset Subscriber Filter
Command Transform
Subscriber
Placement
Merge Processor
Creation
No
Yes
Match?
Matching
No
Yes
Add?
Association Processor
Event Sequencer
Event Transform
TAO File
Driver's Event Cache
Sync and Ignore Subscriber Filter
Not part of the channel thread

A driver is an application shim combined with policies that allows Identity Manager to communicate with an external application in order to synchronize data between the application and the Identity Vault. Note that the term driver and shim are interchangeable. In Figure A-1, the shim is located at the top, linked to an external application and the Identity Vault. Between the driver shim and the Identity Vault are the rules which manage the data.

Data flows through an Identity Manager system in the form of XML documents. Identity Manager has a vocabulary of XML named XDS which is used to represent the state of objects and data operations with the corresponding attribute values.

The Identity Manager engine uses the shim to deliver and consume information with a connected system. It uses the driver configuration rules to decide how and what to do.

Drivers connect to the applications in order to manage objects and entities. A driver has two basic responsibilities:

- Report data changes (events) in the application to the Identity Manager engine.
- Carry out data changes (commands) submitted by the Identity Manager engine to the application.

The combination of a connected system driver, application connection information, and a set of policies is referred to as a driver configuration. Driver configurations are stored in a set of directory objects in the Identity Vault. The DirXML-Driver object contains other objects that define the policies and parameters associated with the configuration.

The driver configuration defines a data pipeline between a connected system and the Identity Vault. The driver configuration defines what might be synchronized and how to map eDirectory schema to a connected system schema or metadata. For example, in an HR application, a user's first name might be referred to as First Name and Given Name in the Identity Vault. In the namespace of the connected system, you refer First Name, but in the name space of the Identity Vault you refer Given Name. In Identity Manager, most of the time you work with the attribute names in the Identity Vault namespace.

A relationship is established between an Identity Vault object and an connected system object when the two objects represent the same entity. This relationship is called an association and is stored in the Identity Vault on the associated Identity Vault object. The association establishes a relationship between the Identity Vault object and the object in the connected system. Key values that uniquely identify objects in connected systems include Global Unique Identifiers (GUIDs), DNs, primary keys in databases, and so on. Each driver is coded to use a specific key.

The Shim

A shim is compiled code that handles translating commands and data between the connected system and Identity Manager.

The driver shim is often written in Java, which uses native application programming interface (API) calls that the system makes available to developers. APIs can include LDAP standard calls, native Windows Active Directory calls, and JDBC connections for SQL databases. The shim has the following responsibilities:

- Translating what the application understands to a standard XML document
- Creating and maintaining the connection to the connected application

- Managing the commands sent from Identity Manager and the connected application
- Monitoring the connected application for changes

For example, if the connected system is an HR system, and a new person is hired, the shim needs to build an XML document that describes this information. In Identity Manager terminology, this is an Add event and an XML document is built to describe this event to the Identity Manager engine. The event is submitted to the engine and a new user is created in the specified location.

After the new user object is created in the Identity Vault, an event is generated for other drivers that monitor changes to user objects. For example, if you have the GroupWise driver deployed, an Add event is generated for the GroupWise driver to create an e-mail box for the new user.

Channels

The flow of data between the Identity Vault and a connected system has two directions named Publisher and Subscriber. These directions are named from the point of view of the connected system:

- The Subscriber channel is the channel in which data flows from the Identity Vault to the application via the shim. The applications subscribe to data from the Identity Vault.
- The Publisher channel is the channel in which data flows from the application to the Identity Vault. The applications publish data to the Identity Vault.

There are cases in which policy might cause data to conceptually flow backward in a channel. This is referred to as channel write-back.

Events and Commands

The distinction between Events and Commands is subtle but important. The report of a change in data at the channel input is an event. Events occur both in the Identity Vault and in the connected system. Examples of events include:

- Creation of an object
- Modification of object attribute values
- Changing of an object's name
- Movement of an object within the object hierarchy
- Deletion of an object

An event coming from the Identity Vault sent over the Subscriber channel is eventually turned into a command to be submitted to the driver shim to cause some change in the connected system. An event coming from the application sent over the Publisher channel is eventually turned into a command to be submitted to the Identity Vault to synchronize the change that occurred in the application.

Commands are the output of a driver channel. When the shim sends an event notification to Identity Manager, the shim is informing Identity Manager of a change in data that occurred in the connected system. Identity Manager then determines, based on configurable policies, which commands, must

be sent to the Identity Vault. When Identity Manager sends a command to the shim, Identity Manager has already taken an Identity Vault event as input, applied the appropriate policies, and determined that the change in the connected system represented by the command is necessary.

From the point of view of the overall system, if a command from one driver on its Publisher channel is creating or updating an object in the Identity Vault, it might cause events to be submitted on the Subscriber channels of other drivers in the system. This allows changes to cascade, flowing to all connected systems.

Schema Mapping Policy

The Schema Mapping policy applies to both the Subscriber channel and to the Publisher channel. The purpose of the Schema Mapping policy is to map schema names, particularly attribute names and class names, between the Identity Vault namespace and the connected system namespace. The Schema Mapping policy is applied before the Output Transformation policy when Identity Manager submits or returns a document to the shim and after the Input Transformation policy when the driver submits or returns a document to the Identity Manager.

Referring to the example of a new hire used earlier, the HR system uses First Name and the Identity Vault uses Given Name, when both refer to the same attribute. The Schema Mapping policy handles the change in names between the connected system's namespace and the Identity Vault's namespace.

The Schema Mapping policy is bidirectional. It overlaps both channels. On the Publisher channel, the connected system's names are mapped to the Identity Vault. On the Subscriber channel, the Identity Vault names are mapped to the connected system.

Event Transformation Rule

The Event Transformation rule operates on events reported on a channel input. The Subscriber and Publisher channels usually have different Event Transformation rules. The purpose of the Event Transformation rules is to modify the report of the events before the events are processed further by Identity Manager. Note that Merge operations do not transit the Event Transformation rule.

There are many common applications for the Event Transformation rules, including:

- Scope filtering (for example, only allow events on objects in a particular subtree, or with a particular attribute value)
- Custom event filtering (for example, disallow moves or deletes)
- Transforming the event directly into a custom command to be passed to the connected system
- Generating additional events

Publisher

The input to the Publisher channel is a description of an event coming from the connected system. The purpose of the Event Transformation rule is to modify that event description. This is applied after the Input Transformation policy and Schema Mapping policy, but before any other policy-based

event processing. The policies implemented in the Event Transformation rule act on the event, such as Add, Delete, or Modify, and not on the data in the event. This is the place where policies are applied to events. For example, you can apply a policy that blocks add events.

If an Add operation is converted into a Merge operation, the current document is discarded, and the filter is used to query to both the connected system and to Identity Vault for all values. The setting for each attribute in the filter is used to decide what to do with the data. The options include overwriting the source information with the information from the destination, overwriting the destination with the source, combining the two and updating both with the results, or doing nothing.

If an Add event contains an association value, the Identity Manager engine turns it into a Modify event.

Subscriber

The input to the Subscriber channel is a description of an event coming from the Identity Vault. In many cases, the filter might be used to determine the types of objects you want, and the attributes of those objects, but the Event Transformation policy can be used to further customize the events. This can be referred to as scope filtering, and it allows for much finer control of what gets through.

For example, you can use filter to specify user objects. It assumes that you want all users synchronized. If a connected system is limited to a subset of all users, then the Event Transformation policy is used to decide if an event for an object is in scope or not. For example, if your connected system should have only users with a department attribute of Sales in it, then a rule on the Event Transformation policy to block any event that is for a user that does not have Sales as its department can accomplish this goal.

Filter

The filter controls the flow of data between the Identity Vault and the connected system. The filter plays several roles in an Identity Manager driver configuration. Figure A-1 shows filter in four places representing most of its roles, but there is really only one filter for the driver.

The driver filter specifies the classes of objects and the attributes of those objects for which Identity Vault processes events and commands for both channels. The filter instructs the Identity Manager engine about events and information the driver's configuration is interested in. From the Identity Vault side, events are queued for the driver if they match an object class in the filter, and if they match an attribute that is set to Sync, Notify, or Reset. Events that occur in the Identity Vault that do not match the data types specified in the filter are ignored by this driver. Similarly, for the application, events that occur that do not match the data types specified in the filter are ignored, though the shim might still need to examine them to see if they need to be handled. For example, if the Identity Manager driver configuration should synchronize only user information, the filter specifies User objects and modification to other Identity Vault objects is ignored. From the possible User class attributes, the filter specifies the selected attributes, such as CN, Given Name, Surname, and Telephone Number. Modifications to other user class attributes is ignored.The user object class and set of related data attributes are listed in the filter for most connected systems.

While the channels allow for data flow, policies and filters are placed in the channel to regulate what gets through and how it looks when it reaches the destination. For example, by configuring the driver filter you can block an attribute value, such as a telephone number from reaching the Identity

Vault from the connected system or vice versa. This helps to regulate whether the Identity Vault or the connected system is the authoritative source to meet specific business requirements. For example, if the filter for the relationship between the PBX system and the Identity Vault allows an employee's telephone number to flow from the PBX system into the Identity Vault but not from the Identity Vault to the PBX system, then the PBX system is the authoritative source for the telephone number. If all other connected system relationships allow the telephone number to flow from the Identity Vault to the connected systems, but not vice versa, the net effect is that the PBX system is the only authoritative source for employee telephone numbers in the enterprise.

- "The Sync Attribute" on page 221
- "The Notify Attribute" on page 221

The Sync Attribute

On the Publisher channel, when an event has been queued for the channel to process and it has passed through the Input Transformation rule, the Schema Map, and the Event Transform, the Sync attributes are selected from the input document, and any attributes not set to Sync or Notify are removed. Attributes that are set to Reset are also handled by querying Identity Vault for the correct value, and having the correct value sent back to the connected system to undo the change that has just been made.

On the Subscriber channel, the Sync filter works the same way it works for the Publishes channel. The only difference is that events are coming from the Identity Vault instead of the connected system.

The Notify Attribute

Notify is a way for attribute data to be used in the event document, without it actually being synchronized to the Identity Vault. For example, you need a person's first name, middle name, and last name from your HR system in order to create an account, but you do not actually want to store the middle name in the Identity Vault. By setting the middle name attribute to Notify, you can access the attributes value without having to store it in the Identity Vault. Any attributes set to Notify are stripped out of the document prior to being submitted to the destination.

Add Processor

- "Publisher" on page 221
- "Subscriber" on page 222

Publisher

The Add Processor is used to decide if an event is an add document. This is a branching point in the driver's processing of the event. An add document is redirected to the Matching rule. The shim supplies the association value, allowing the Identity Manager engine to quickly and easily find the correct object in the Identity Vault. Associations are created as a match between two objects or when an object is newly created in either the Identity Vault or the connected system. After an association is formed between objects, this association remains in effect until the objects are

deleted or the association is deleted by the administrator. Well-designed Matching rules automate the creation of associations between existing objects in the Identity Vault and the connected system. For more information, see "Associations" on page 228.

If it is not an add document, it moves on to the Command Transformation as the next step.

Subscriber

On the Subscriber channel, the Add Processor is used to decide if an event is an add document. This is a branching point in the driver's processing of the event. An add document is redirected to the Matching rule. The Identity Manager engine uses the association value of an Identity Vault object to allow the shim to modify the correct object in the connected system. For more information, see "Associations" on page 228.

If it is not an add document, it moves on to the Command Transformation as the next step.

When a Modify event does not contain an association that resolves to an actual object when it arrives at the Add Processor, the Identity Manager engine attempts to create it. This is the Synthetic Add process, and it can happen on either the Publisher or the Subscriber channel.

The Identity Manager engine uses the Modify event to figure out which object to work with. It then uses the filter to query back to get all attributes that are available for that object that are set to Sync or Notify on the current channel. It discards the Modify event and builds an Add event to replace it. The Add event is forwarded through the Matching, Create, Placement, and Command Transform rules (on the Subscriber it also goes through the Schema Map and Output Transform). The Event Transformation rule is not applied for Synthetic Adds. This is due to the rule location before the Add Processor.

Matching Rule

Matching rules establish links between an existing object in the Identity Vault and an existing object in the connected system. The matching rules specify which class and attribute values must match for an object in the Identity Vault and an object in the connected system to be marked as corresponding entries.

A good matching rule requires you to investigate both systems involved, and find the data that guarantees a 1:1 mapping between them. Attributes such as employee ID number, email address, and badge number are some of the more common pieces of data used for matching criteria. If there is no single attribute available, the combinations of attributes might be used. Matching on Surname only is not a good criteria. For example, in larger organizations, there might be a possibility that two employees have the same last name. Matching on Surname + Given Name would produce higher quality matches and matching on Surname + Given Name + Department would further increase the probability of correct matching. If a match is successful, an association between the two objects is created. If a match is not successful, the Create rules are used.

- "Publisher" on page 223
- "Subscriber" on page 223

Publisher

The Matching rule is used to link an object in the Identity Vault with the corresponding object in the connected system. For example, if you are connecting an existing HR system to an existing eDirectory system, there are people in the HR system, and users in the Identity Vault, and they both represent the same user. The Matching rule contains rules which allow Identity Manager to determine that ”Joe Doe” in HR system is ”jdoe13” in the Identity Vault.

The Matching rule uses matching criteria and queries Identity Vault looking for a matching object. The Matching rule returns zero when no object is matched, so that the Add event continues to be processed. It returns one when one matching object is found, which means that the object in the input document matches an object in the Identity Vault. After the objects are matched, the data between the two objects is merged based on filter settings. If the Matching rule finds more than one matching object, the Identity Manager engine treats this as an error and quits the transaction. You should either modify the Matching rule or manually handle this conflict.

Subscriber

On the Subscriber channel, the Matching rule works on the Add events and uses the Identity Vault data to query the connected system looking for matching objects.

Create Rule

The Create rules are applied to the Add events when the Matching rules fail to find a match. The Create rules specify the minimum set of data that an event must have before an object can be created in the Identity Vault or the connected system.

- “Publisher” on page 223
- “Subscriber” on page 224

Publisher

If the Matching rule does not find a matching object in the Identity Vault, the Create rule is applied to the document to ensure that the document contains sufficient information. It is also used to supply default values for attributes, and it might specify a template to be used in the creation of the new object. From the Identity Vault side, a user object must have a name (CN or UID) and it must have a Surname. While this might be enough for Identity Vault objects to be created, most organizations might need additional information before creating an account. The driver can reject documents that do not contain sufficient information to continue processing.

The Create rule can also veto an Add event if the Add event fails to meet the conditions imposed by the Create rule. For example, if the Create rule requires an object to have a telephone number and it doesn't have one, the Add event is vetoed.

The discarded events are reprocessed when the additional attribute information is added in the connected system. This results in a Modify event without an associated object, which the Add Processor converts to a Synthetic Add.

Subscriber

The Create rule in Subscriber works the same as the Publisher channel in determining if an event has sufficient information to create an object in the connected system. This requires knowledge of the connected system and its technical or business requirements.

The Create rule is often used to examine the attributes available for the new object (from the source event) and vetoes the creation of the new object if one or more required attributes is missing. The most common example on the Subscriber channel is to require a password. Normally a user is created in the Identity Vault in two stages, first as a user object and then as a second operation a password is set. It is very common to see that the object is created, but the Create rule fails due to the lack of password which is a required attribute for creating a new user object. A moment later when the password event comes through, the new object is successfully added.

Placement Rule

- "Publisher" on page 224
- "Subscriber" on page 224

Publisher

If the Matching rule determines that there are no matching objects, and the Create rule verifies that the event meets the minimum requirements, the Placement rule specifies which object to create, and where to place. For the Publisher channel, the object created is placed in the Identity Vault using the naming rules contained in policy. For example, the Identity Vault places all objects in the `Data\Users\` container.

Subscriber

On the Subscriber channel, the Placement rule works the same as the Publisher channel. Placements in connected systems can be simple or complex. A simple Placement rule places all objects in the same location.

Placements in the connected systems require a detailed knowledge of how objects are represented in the connected system.

A more complex example might use an HR location code attribute to determine where to place an object. You can use a Mapping Table to help establish the relationship between a placement location and an attribute value.

Command Transformation Rule

The Command Transformation rule operates on commands that are about to be issued to a channel output. The Subscriber and Publisher channels usually have different Command Transformation rules. The purpose of the Command Transformation rule is to provide final processing on commands before the commands are sent to the Identity Vault or to the connected system.

Some possible applications for the Command Transformation rule include:

- Changing the command type (for example, an object delete command might be transformed into a modification that will cause the object to be archived)
- Blocking commands
- Adding additional commands
- Controlling the output of the Identity Manager engine's Merge process
- "Publisher" on page 225
- "Subscriber" on page 225

Publisher

All events pass through the Command Transformation rule. This is where the earlier branch at the Add Processor rejoins the flow. Most of the driver policies reside in the Command Transformation, because conceptually this is where the conversion from event to command happens. Up to this point, the document has been describing an event that has happened in the connected system. Now that event is converted into a command and applied to the Identity Vault. It is the last chance to modify a command before it is applied to the Identity Vault.

Subscriber

On the Subscriber channel, most of the driver policies reside in the Command Transformation, because conceptually this is where the conversion from event to command happens and before the Schema Mapping policy is applied. Both the Schema Mapping policy and the Output Transformation policy are executed after the Command Transformation policy on the Subscriber channel. Up to this point, the document has been describing an event that occurred in the Identity Vault. Now this event is converted into a command and applied to the connected system.

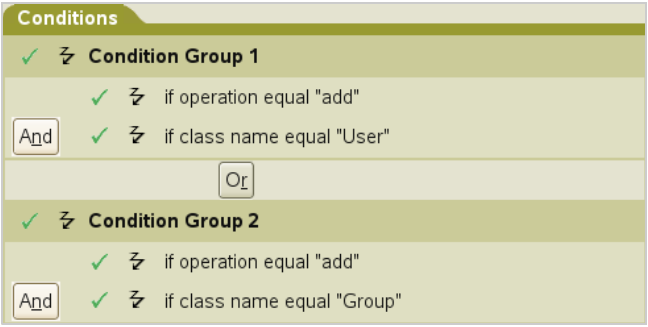
Rules, Policies, and Style Sheets

Within any one of the rules covered above (Input Transform, Command Transform, etc.) are zero or more policies. Some of these may come from the preconfigured driver import used as a starting point. Others could be customizations of the driver configuration.

A rule in Identity Manager is a collection of policies. Each rule has conditions that must be met, and actions to be carried out when the conditions are true. The grammar of the conditions is meant to be human readable and generally to make sense. For example, a condition of "if object class equal user" would be True if the object being described in the current document is a User object, and would be False if the object is a Group. Conditions are made up of Condition Groups. Within a Condition Group, all of the conditions can be combined with And or Or, and the result must be True

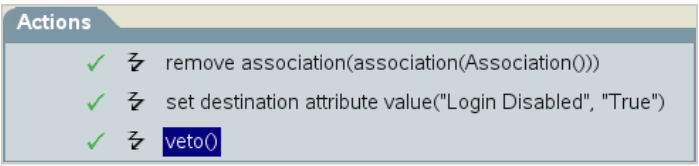
for the Condition Group to evaluate to True, otherwise it evaluates to False. Multiple Condition Groups can also be combined, using And and Or. After the Condition Group(s) have evaluated to True, the Rule's Actions are performed.

Figure A-2 *Policy Builder Conditions*



Actions can act on the current document, and in many cases this is sufficient. But Actions can also query the source or destination, which could be Identity Vault or the connected system, depending on which channel you are on, for additional information. Actions can change the current document in to a modified version of itself, or can block it entirely. Actions can be used to make different documents. A document describing a delete event in a connected system could be tested by a Condition (if operation equal delete), and acted on by a set of Actions to prevent the associated object from being deleted in Identity Vault (veto), but to modify that object to remove its association value for this application (remove-association), and to disable it (set destination attribute value Login Disabled = True).

Figure A-3 *Policy Builder Actions*



Policies define what data is transferred and how the data is synchronized between the connected system and the Identity Vault. A default set of policies is available with the driver configuration. Other policies might be local customizations of the driver. You can write policies using the DirXML Script, XSLT, or ECMA Script.

The purpose of a policy is to make changes to the input document and produce an output document. Most policies are evaluated either on the Subscriber channel or on the Publisher channel. The Schema Mapping policy, the Input Transformation policy, and the Output Transformation policy are evaluated on both channels. For example, one organization might use the `inetOrgPerson` as the main user class, while another organization might use User. A policy can be implemented to add the phone number change to an `inetOrgPerson` for the first organization, and a separate rule can be implemented to make it work for the User class. Policies make schema transformations, specify matching criteria to determine if an object already exists in the connected system or the Identity Vault, and many other things. Because of this, an Add event reported by your connected system might end out as a Modify operation in the Identity Vault, if a matching policy determines that the object you added already exists in the Identity Vault.

On the Subscription channel, when a new user is created in the Identity Vault and you want it created in the connected system, before sending this command to the driver, the Identity Manager engine calls a series of policies. These policies define the way objects are created and determine if a corresponding user already exists in the connected system, make decisions about placement, provide default values for required attributes that are not specified, and so on. This Add event might be transformed into a Modify event if the object exists in the connected system. Attributes that were not contained in the original event could be added to conform with the object creation model of the connected system.

Style sheets define XSLT transformation rules. Style sheets transform input or output commands into a different command, change an event from one type to another, or perform other arbitrary XML transformations. For more information, see Identity Manager Style Sheets.

- "Input Transform Rule" on page 227
- "Output Transform Rule" on page 228
- "Associations" on page 228
- "Synthetic Adds" on page 229
- "Merge Processing" on page 231

Input Transform Rule

The Input Transformation rule applies to both the Subscriber channel and to the Publisher channel. The purpose of the Input Transformation rule is to perform a preliminary transformation on all XML documents sent to Identity Manager by the connected system and returned to Identity Manager from the connected system. The Input Transformation rule is applied to the XML documents sent to `XmlCommandProcessor.execute` and `XmlQueryProcessor.query` when called by the connected system and to the XML documents returned from `SubscriptionShim.execute` and `XmlQueryProcessor.query` when called by Identity Manager engine. The Input Transformation policy is applied before the Schema Mapping policy.

The Input Transform rule is often used to transform data from the application format into the Identity Vault format. When the Input Transformation is used for data format transformations the Output Transformation policy usually performs the data transformation in the opposite direction (transforms data from the Identity Vault format to the connected system format). This rule operates in the connected system's (application's) namespace. For example, it might be used to reformat data, such as changing a phone number that is formatted as 1(815)555-1212 to 1-815-555-1212.

The Input Transformation rule is also used to perform actions in response to the results of commands sent to the shim. Note that the schema names are always in the application namespace in the XML processed by the Input Transformation policy.

It is also possible to use the Input Transformation rule to transform an arbitrary XML format native to the connected application to the format expected by Identity Manager. Such transformations must be written in XSLT because DirXML-Script operates only on the Identity Manager specific XML vocabulary that is specific to Identity Manager. A few examples are the Delimited Text driver and the SOAP driver.

Output Transform Rule

The Output Transformation policy applies to both the Subscriber channel and to the Publisher channel. The purpose of the Output Transformation policy is to perform a final transformation on all XML documents sent to the shim by the Identity Manager engine and returned to the shim by Identity Manager engine. The Output Transformation policy is applied to the XML documents sent to `SubscriptionShim.execute` and `XmlQueryProcessor.query` when called by the Identity Manager engine and to the XML documents returned from `XmlCommandProcessor.execute` and `XmlQueryProcessor.query` when called by the shim. The Output Transformation rule is applied after the Schema Mapping rule.

The Output Transform rule is the converse of the Input Transform rule. It modifies the command that is about to be submitted to the shim as required. This usually involves undoing what has been done in the Input Transform rule. If you have an Input Transform rule that converts phone numbers formatted as 1(815)555-1212 to 1-815-555-1212, you need to have an Output Transform rule that converts 1-815-555-1212 to 1(815)555-1212.

You can also use the Output Transformation policy to transform the format used by Identity Manager to an arbitrary XML format native to the connected application. These transformations must be written in XSLT because DirXML-Script operates only on the XML vocabulary that is specific to Identity Manager.

Associations

The Association value is Identity Manager's way of keeping track of which object in the connected system matches an object in the Identity Vault. Each driver handles this slightly differently. In almost all cases, this should be a 1:1 match, so that it is possible to say that "john doe", employee number 1234567 in the HR system matches exactly with the user object "jdoe13" in the Identity Vault, with "doe john" in Active Directory, and jdoe13@example.com" in the e-mail system. Most connected systems have some sort of internal unique identifier, even if it is not the one that you usually see in the system's management tools. eDirectory and Active Directory have a globally unique identifier or GUID. Many HR systems have an employee number. E-mail systems usually have a unique email address value for each person. Identity Manager uses these identifiers to build its association.

Associations are stored in the Identity Vault only. On the Subscriber channel, the Identity Manager engine uses this value to allow the shim to modify the correct object in the connected system. On the Publisher channel, the shim supplies the association value, allowing the Identity Manager engine to quickly and easily find the correct object in the Identity Vault to work with. The following association states are stored in the Identity Vault:

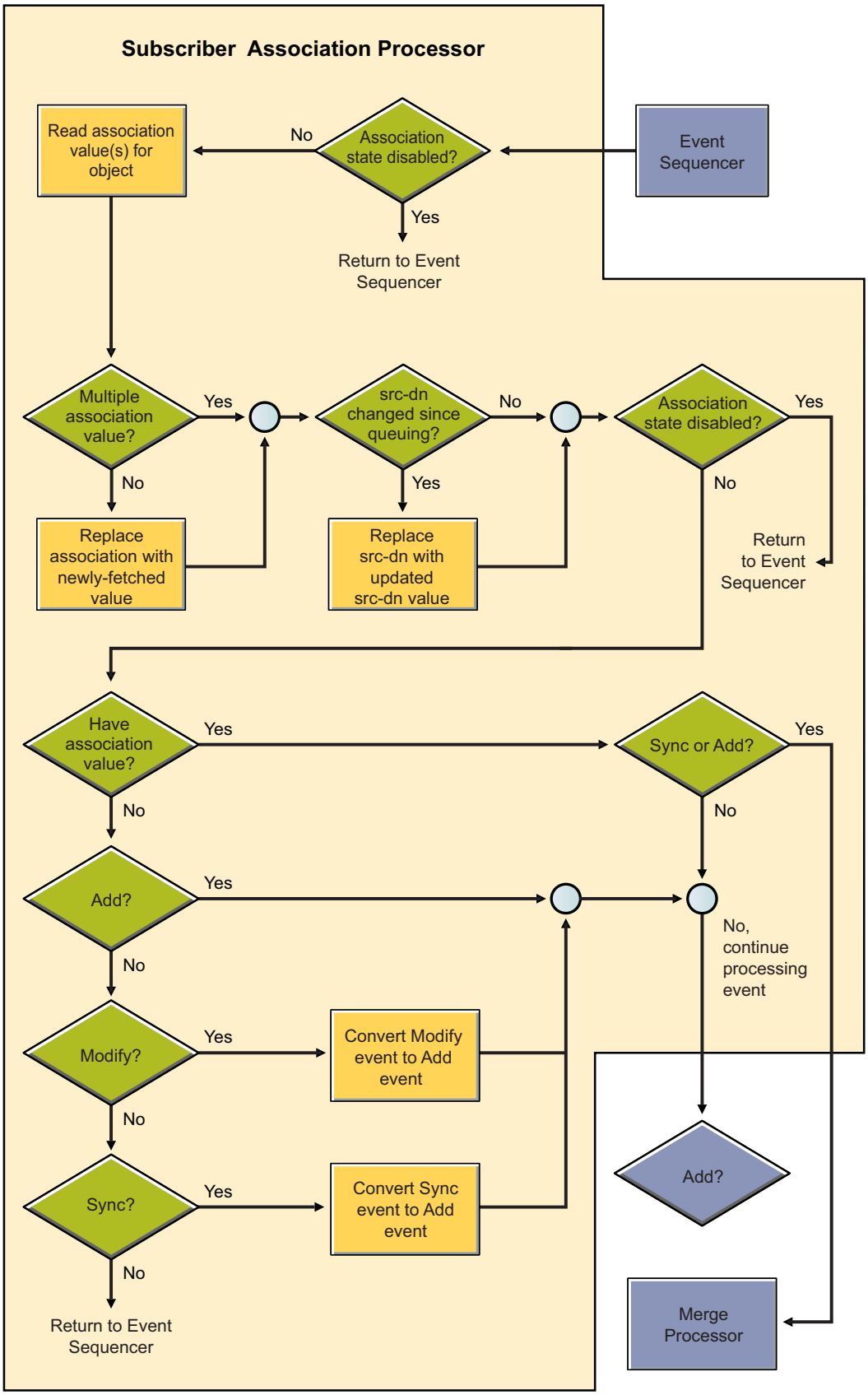
- **0 Disabled**: Changes in the driver objects are not synchronized with the Identity Vault.
- **1 Processed**: Successful association has been created between the driver objects and Identity Vault.
- **2 Pending**: The Identity Manager engine identified a modification to an object, and attempted to match it or create it in the connected system, but was unable to do so.
- **3-Manual**: A manual association was created by the user.
- **4-Migrate**: The account was synchronized or migrated.
- **blank No association**: No association has been created.

Synthetic Adds

When a Modify document without an association encounters the Add Processor, the Identity Manager engine converts the Modify event into a Synthetic Add process. This process occurs in the Publisher or the Subscriber channel.

The Identity Manager engine uses the Modify event to figure out which object to work with. It then uses the filter to query back to get all attributes that are available for that object that are set to Sync or Notify on the current channel. It then throws away the Modify document and builds an Add document to replace it. This Add document is then forwarded through the Matching, Create, Placement, and Command Transformation (on the Subscriber it also goes through the Schema Map and Output Transformation).

Figure A-4 Subscriber Channel Association Processor



Merge Processing

A Merge operation occurs when the Identity Manager engine converts an Add operation into a Modify operation. This happens most commonly during an initial migration, as the migration sends objects down a channel, and the Matching rule finds an object that it can use to associate with the object being migrated.

In a Merge operation, the current document is discarded again (like the Synthetic Add), and the filter is used to query both the connected system and the Identity Vault for all values. The setting for each attribute in the filter is used to decide what to do with the data. The options include overwriting the

source information with the information from the destination, overwriting the destination with the source, combining the two and updating both with the results, or doing nothing. The following flow charts illustrate the Publisher Merge Processor and the Subscriber Merge Processor.

Figure A-5 *Publisher Merge Processor*

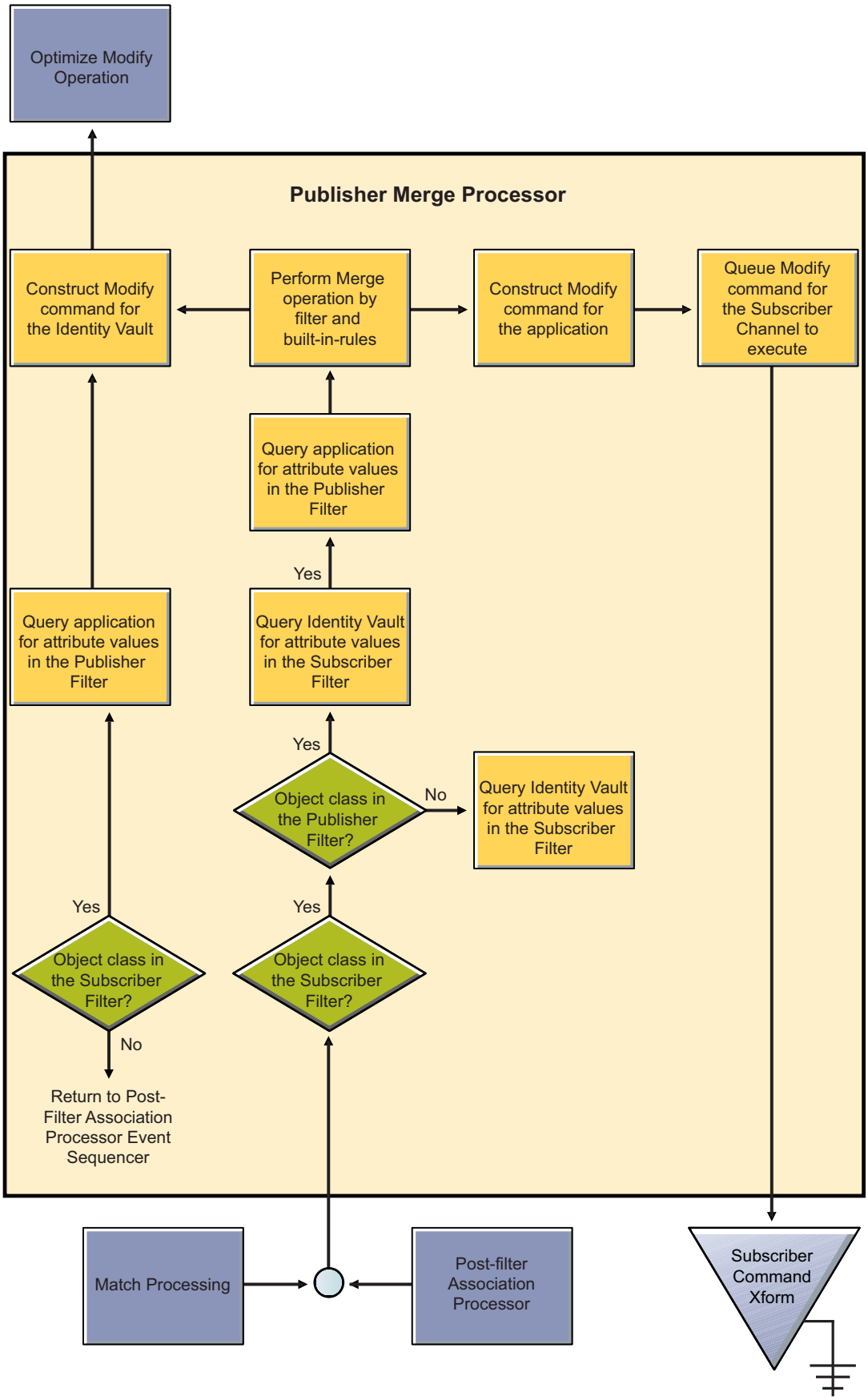
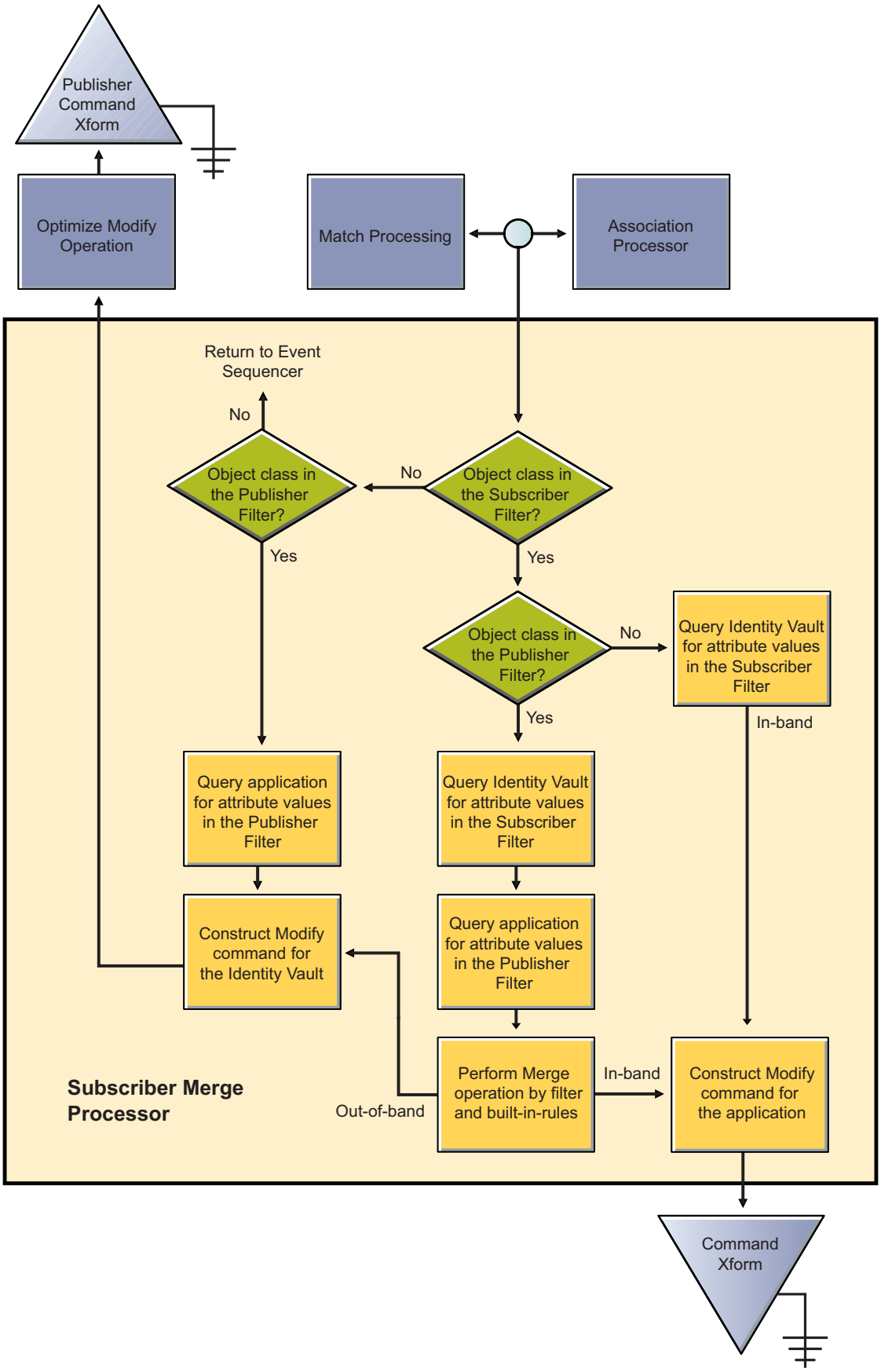

Optimize Modify Operation
Publisher Merge Processor
Construct Modify command for the Identity Vault
Perform Merge operation by filter and built-in-rules
Construct Modify command for the application
Queue Modify command for the Subscriber Channel to execute
Query application for attribute values in the Publisher Filter
Yes
Query Identity Vault for attribute values in the Subscriber Filter
Query application for attribute values in the Publisher Filter
Yes
Object class in the Publisher Filter?
No
Query Identity Vault for attribute values in the Subscriber Filter
Yes
Object class in the Subscriber Filter?
Yes
Object class in the Subscriber Filter?
No
Return to Post-Filter Association Processor Event Sequencer
Match Processing
Post-filter Association Processor
Subscriber Command Xform

Figure A-6 *Subscriber Merge Processor*


Publisher Command Xform
Optimize Modify Operation
Match Processing
Association Processor
Return to Event Sequencer
No
Object class in the Publisher Filter?
No
Object class in the Subscriber Filter?
Yes
Yes
Object class in the Publisher Filter?
No
Query Identity Vault for attribute values in the Subscriber Filter
Yes
In-band
Query application for attribute values in the Publisher Filter
Query Identity Vault for attribute values in the Subscriber Filter
Construct Modify command for the Identity Vault
Query application for attribute values in the Publisher Filter
Subscriber Merge Processor
Perform Merge operation by filter and built-in-rules
In-band
Construct Modify command for the application
Out-of-band
Command Xform

B Driver Properties

This section provides information about the properties that are common to all drivers. This includes all properties (Named Password, Engine Control Values, Log Level, and so forth) other than the Driver Configuration and Global Configuration Values properties.

The information is presented from the viewpoint of Identity Console. If a field is different in Designer for Identity Manager, it is marked with a Designer icon.

Driver Configuration

In Identity Console:

1. Click the **Drivers** tab from the Identity Console home screen.
2. Click the driver name link to see the driver's configuration page.

   By default, the Connection Parameters page appears.

In Designer:

1. Open a project in the Modeler.
2. Right-click the driver icon or line, then select click **Properties > Driver Configuration.**

The Driver Configuration options are divided into the following sections:

- "Driver Module" on page 237
- "Authentication" on page 238
- "Startup Option" on page 239
- "Driver Parameters" on page 239
- "ECMAScript" on page 240
- "Global Configuration" on page 240

Driver Module

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

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

For example, `com.microfocus.nds.dirxml.driver.scim.SCIMDriverShim`

Native: This option is used to specify name of the `.dll` which is developed in native language (such as C++) for the driver.

For example, `addriver.dll`

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

- **Remote Loader Connection Parameters**: Includes information of the Remote Loader environment details such as, Host Name, Connection Port, etc.
- **Remote Loader Password**: The password for the Remote Loader.
- **Driver Object Password**: Specifies a password for the Driver object. If you are using the Remote Loader, you must enter a password on this page. The Remote Loader uses this password to authenticate itself to the remote driver shim.

Authentication

This section describes the parameters required for server authentication for Identity Manager Engine and the Remote Loader.

Field	Description
Authentication information for server	Displays or specifies the IP address or server name that the driver is associated with.
Application authentication	◆ **Authentication ID**: Specify a user application ID. This ID is used to pass Identity Vault subscription information to the application. ◆ **Connection Information**: Specify the IP address or name of the server the application shim should communicate with. ◆ **Set Password**: Option to set the application authentication password.
Remote Loader authentication (Optional): Used only if the driver is connecting to the application through the Remote Loader.	◆ **Host name**: Enter the Host Name or IP Address where the Remote Loader Service has been installed and is running for this driver. ◆ **Port**: Enter the Port Number where the Remote Loader Service has been installed and is running for this driver. The Default Port is 8090. ◆ **KMO**: Enter the Key Name of the Key Material Object containing the keys and certificate used for SSL The KMO parameter is used only when an SSL connection exists between the Remote Loader and the Metadirectory engine. ◆ **Other parameters**: Enter other parameters required, if any, in the connection string. These parameters MUST be in the key-value pair. Example, `paraNamel =paraValue1 paraName2=paraValue2` ◆ **Set Password**: 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.

Startup Option

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

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

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

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

Driver Parameters

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

The parameters are presented by category:

- "Driver Settings" on page 239
- "Subscriber Settings" on page 239
- "Publisher Options" on page 240

Driver Settings

Custom Java Extensions: Select **Show** if you have developed custom Java classes to extend the driver shim's functionality. Otherwise, select **Hide**.

- **Document Handling:** Select **Implemented** if you have developed a custom Java class to process data as XML documents. Otherwise, select **None**.
  - **Class:** Specify the class by using a complete package identifier. For example, `com.novell.DocumentModifier`.
  - **Init Parameter:** Specify the parameter to pass to the init() method of the specified class. The init method is responsible for parsing the information contained in this string. Leave this field blank if the configuration string is not required for the class.

Subscriber Settings

Authentication Method: The authentication types are configured based on the parameters that the connected application supports. For more information about the authentication types the driver supports refer to the specific driver guide on the Identity Manager Drivers Documentation Website (https://www.netiq.com/documentation/identity-manager-49-drivers/).

Publisher Options

After the driver is configured, you can edit the publisher options to specify Polling Interval, Polling Resources and Heartbeat Interval. Navigate to **Driver Configuration > Publisher Options** tab to edit or specify these options. For more information refer to the specific driver guide on the Identity Manager Drivers Documentation Website (https://www.netiq.com/documentation/identity-manager-49-drivers/).

ECMAScript

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

Global Configuration

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

Named Passwords

Identity Manager enables you to securely store multiple passwords for a driver set or individual drivers. 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, such as a user name.

Engine Control Values

The engine control values are a way that certain default behaviors of the Identity Manager engine can be changed. The values can be accessed only if a server is associated with the Driver Set object.

Option	Description
Subscriber channel retry interval in seconds	The Subscriber channel retry interval controls how frequently the Identity Manager engine retries the processing of a cached transaction after the application shim's Subscriber object returns a retry status.
Qualified form for DN-syntax attribute values	The qualified specification for DN-syntax attribute values controls whether values for DN-syntax attribute values are presented in unqualified slash form or qualified slash form. A True setting means the values are presented in qualified form.

Option	Description
Qualified form from rename events	The qualified form for rename events controls whether the new-name portion of rename events coming from the Identity Vault are presented to the Subscriber channel with type qualifiers. For example, CN=. A True setting means the names are presented in qualified form.
Maximum eDirectory replication wait time in seconds	This setting controls the maximum time that the Identity Manager engine waits for a particular change to replicate between the local replica and a remote replica. This affects only operations where the Identity Manager engine is required to contact a remote eDirectory server in the same tree to perform an operation and might need to wait until some change has replicated to or from the remote server before the operation can be completed (for example, object moves when the Identity Manager server does not hold the master replica of the moved object; file system rights operations for Users created from a template.)
Use non-compliant backwards-compatible mode for XSLT	This control sets the XSLT processor used by the Identity Manager engine to a backwards-compatible mode. The backward-compatible mode causes the XSLT processor to use one or more behaviors that are not XPath 1.0 and XSLT 1.0 standards-compliant. This is done for backward compatibility with existing DirXML style sheets that depend on the non-standard behaviors. For example, the behavior of the XPath “!=” operator when one operand is a node-set and the other operand is other than a node-set is incorrect in DirXML releases up to and including Identity Manager 2.0. This behavior has been corrected; however, the corrected behavior is disabled by default through this control in favor of backward compatibility with existing DirXML style sheets.
Maximum application objects to migrate at once	This control is used to limit the number of application objects that the Identity Manager engine requests from an application during a single query that is performed as part of a Migrate Objects from Application operation. If java.lang.OutOfMemoryError errors are encountered during a Migrate from Application operation, this number should be set lower than the default. The default is 50. **NOTE:** This control does not limit the number of application objects that can be migrated; it merely limits the batch size.
Set creatorsName on objects created in Identity Vault	This control is used by the Identity Manager engine to determine if the creatorsName attribute should be set to the DN of this driver on all objects created in the Identity Vault by this driver. Setting the creatorsName attribute allows for easily identifying objects created by this driver, but also carries a performance penalty. If not set, the creatorsName attribute defaults to the DN of the NCP Server object that is hosting the driver.
Write pending associations	This control determines whether the Identity Manager engine writes a pending association on an object during Subscriber channel processing. Writing a pending association confers little or no benefit but does incur a performance penalty. Nevertheless, the option exists to turn it on for backward compatibility.

Option	Description
Use password event values	This control determines the source of the value reported for the nspmDistributionPassword attribute for Subscriber channel Add and Modify events. Setting the control to False means that the current value of the nspmDistributionPassword is obtained and reported as the value of the attribute event. This means that only the current password value is available. This is the default behavior. Setting the control to True means that the value recorded with the eDirectory event is decrypted and is reported as the value of the attribute event. This means that both the old password value (if it exists) and the replacement password value at the time of the event are available. This is useful for synchronizing passwords to certain applications that require the old password to enable setting a new password.
Retry Out of Band events	This control determines whether the out-of-band sync events should be retried or not if the **retry** status for the out-of-band sync event is received. If the control is set to False, the out-of-band sync is not retried. If it is set to true, the out-of-band sync is retried till its successful.
Use Rhino ECMAScript engine	Determines whether the Identity Manager engine uses the Rhino ECMAScript engine. The engine uses Rhino as the default ECMAScript engine. This control is **true** by default, if you set this control to **false** engine uses Nashorm script.
Enable Subscriber Service Channel	Determines whether the Identity Manager engine processes the out of band queries on the Subscriber Service channel of the driver. Some common examples of these queries are code map refresh, data collection, and queries triggered from dxcmd. When this control is set to true, the channel separately processes these queries without interrupting the normal processing of events. Currently, this control is only available for use with the JDBC Fan-Out driver (enabled by default).
Enable password synchronization status reporting	This control determines whether the Identity Manager engine reports the status of Subscriber channel password change events. Reporting the status of Subscriber channel password change events allows applications such as the Identity Manager User Application to monitor the synchronization progress of a password change that should be synchronized to the connected application.
Combine values from template object with those from add operation	This value determines whether the Identity Manager engine combines like values from a creation template and an add operation when performing the add operation. Setting the value to True causes the template's multi-valued attribute values to be used in addition to the values for the same attribute that are specified in the add operation. Setting the value to False causes the values from the template to be ignored if there are values for the same attribute specified in the Add operation.

Option	Description
Allow event loopback from publisher to subscriber channel	This value determines whether the Identity Manager engine allows an event to loop from the driver's Publisher channel to the Subscriber channel. Setting the value to False causes the Identity Manager engine to not allow events to loop back. Setting the value to True causes the Identity Manager engine to allow events to loop from the Publisher channel to the Subscriber channel.
Revert to calculated membership value behavior	This value determines the method used by the Identity Manager engine when performing read and search actions related to group membership. Setting this value to False (the default setting) causes the Identity Manager engine, when reading or searching the Member and Group Member attributes of Identity Vault objects, to return only those values that are "static" values. Static values are objects that received group membership by direct assignment to the group rather than inherited assignment through a nested group. Setting this value to True causes the Identity Manager engine to revert to the method used prior to Identity Manager 3.6. In pre-3.6 versions, the Identity Manager engine's search of the Member and Group Member attributes retrieved all "calculated" values. Calculated values include objects that are either 1) statically assigned membership or 2) dynamically assigned membership by virtue of the nested group hierarchy calculations used by eDirectory. A search of a group's Members attribute returns any objects that were directly assigned to the group or that were assigned membership through a nested group.
Maximum time to wait for driver shutdown in seconds	This setting controls the maximum time that the Identity Manager engine waits for the driver's Publisher channel to shut down. If the driver does not shut down within the specified time interval, the Identity Manager engine terminates the driver.
Regular Expression escape meta-characters	This control determines the meta-characters that will be escaped while expanding the local variable when used in a regular expression context. All characters that need to be escaped must be added as a comma separated list for this control value. If a meta-character is not present in the control value, then it will not be escaped during local variable expansion containing a regular expression. While using this control, ensure the following: ◆ The value is not left empty. By default, it is populated with ***$***. This character is required for local variable expansion. ◆ The value should be a valid comma(,) separated list, otherwise you will encounter errors during policy evaluation. ◆ To escape all meta-characters, specify "\,$,^,.,?,*,+,[,],(,),|" as a value. ◆ If a meta-character need not be escaped, remove that character from the value. ◆ To escape any meta character, specify the meta character followed by a back slash (\).

Option	Description
Ignore Entitlement Changes of other drivers	This control determines whether the Identity Manager engine ignores or processes entitlement changes of other drivers. The default value is True. This means that the driver automatically ignores the entitlement changes of other drivers. If this control is set to False, the entitlement changes of other drivers are cached and processed by this driver.
Allow Entitlement event loopback from cprs to subscriber channel	This control determines whether the Identity Manager engine allows an entitlement event that is generated by a CPRS assignment to loopback to the Subscriber channel of the driver. The default value is False. This means that the event is not looped back to the Subscriber channel. If this control is set to True, the event flows to the Subscriber channel of the driver.

Log Level

Each driver set and each driver has a log level field where you can define the level of errors that should be tracked. The level you indicate here determines which messages are available to the logs. By default, the log level is set to track error messages. (This also includes fatal messages.) To track additional message types, change the log level.

NetIQ recommends that you use Audit or Sentinel for logging and reporting if possible. See the *Administrator Guide to NetIQ Identity Reporting* and *NetIQ Identity Manager - Configuring Auditing in Identity Manager*.

Option	Description
Use log settings from the DriverSet	If this is selected, the driver logs events as the options are set on the Driver Set object.
Log errors	Logs just errors.
Log errors and warnings	Logs errors and warnings.
Log specific events	Logs the events that are selected. Click the icon to see a list of the events.
Only update the last log time	Updates the last log time.
Logging off	Turns logging off for the driver.
Turn off logging to DriverSet, Subscriber and Publisher logs	Turns all logging off for this driver on the Driver Set object, Subscriber channel, and the Publisher channel.
Maximum number of entries in the log (50-500)	Number of entries in the log. The default value is 50.

Security Equals

Use the Security page to view or change the list of objects that the driver is explicitly security equivalent to. This object effectively has all rights of the listed objects.

If you add or delete an object in the list, the system automatically adds or deletes this object in that object's "Security Equal to Me" property. You don't need to add the [Public] trustee or the parent containers of this object to the list, because this object is already implicitly security equivalent to them.

Designer does not list the users the driver is security equals to.

Filter

Launches the Filter editor.

In Designer, the Filter editor is not included with the driver properties.

To access the Filter editor in Designer:

1. In an open project, click the **Outline** tab (Outline view).
2. Select the driver you want to manage the filter for, then click the plus sign to the left.
3. Double-click the **Filter** icon to launch the Filter editor.

Edit Filter XML

Allows you to edit the filter directly in XML instead of using the Filter editor.

In Designer, the XML Filter editor is not included in the driver properties.

To access the XML Filter editor in Designer:

1. In an open project, click the **Outline** tab (Outline view).
2. Select the driver for which you want to manage the filter, then click the plus sign to the left.
3. Double-click the **Filter** icon to launch the Filter editor, then click **XML Source** at the bottom of the Filter editor.

Misc/Trace

Allows you to add a trace level to your driver. With the trace level set, DSTrace displays the Identity Manager events as the Identity Manager engine processes the events. The trace level affects only the driver for which it is set. Use the trace level for troubleshooting issues with the driver when the driver is deployed. DSTRACE displays the output of the specified trace level.

Option	Description
Trace level	Increases the amount of information displayed in DSTRACE. 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.

Option	Description
Trace file	When a value is set in this field, all Java information for the driver is written to the 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.
Trace name	Driver trace messages are prepended with the value entered in this field.
Use setting from Driver Set	This option is available only in Designer. It allows the driver to use the same setting that is set on the Driver Set object.

Excluded Objects

Use this page to create a list of users or resources that are not replicated to the application. NetIQ recommends that you add all objects that represent an administrative role to this list (for example, the Admin object).

Designer does not list the excluded users.

Driver Health Configuration

Driver health monitoring allows you to view a driver's current state of health as either green, yellow, or red, and to define the actions to perform in response to each of these health states. The Driver Health Configuration options are used to configure the health monitoring.

Driver health is discussed in detail in Chapter 6, "Monitoring Driver Health," on page 77.

Driver Manifest

The driver manifest is like a resumé for the driver. It states what the driver supports, and includes a few configuration settings. The driver manifest is created by default when the Driver object is imported. A network administrator usually does not need to edit the driver manifest.

Driver Cache Inspector

The Driver Cache Inspector displays information about the cache file that stores events being processed by the driver. The Driver Cache Inspector is discussed in detail in Chapter 9, "Managing Driver Cache Files," on page 95.

Designer does not include the Driver Cache Inspector.

Driver Inspector

The Driver Inspector displays information about objects associated with the driver. The Driver Inspector is discussed in detail in Chapter 8, “Managing Associations between Drivers and Objects,” on page 89.

Designer does not include the Driver Inspector.

C Understanding Identity Manager Trace

Data flows through the Identity Manager system in the form of XML documents. Identity Manager has a vocabulary of XML named XDS. Identity Manger uses XDS to represent the state of objects and data operations with the corresponding attribute values. Identity Manager uses DirXML Script to handle identity synchronization events. DirXML Script is an XML-based language and consists of conditions, actions, noun, and verb tokens to modify the data exchanged between the Identity Vault and the external data store. DirXML Script takes the XDS document, determines what needs to be done using the conditions, and then builds the document using the actions.

Identity Manager provides several options for capturing information about the background actions that occur when transactions are processed through Identity Manager. The following are the options:

- Capturing a driver set trace
- Capturing a driver trace
- Using eDirectory's ndstrace

The most common method of capturing information of an issue with a specific driver is using the driver trace. The driver set trace is used while troubleshooting issues related to the driver set, such as issues related to Identity Manager jobs. ndstrace is an all-encompassing trace tool that can be used for tracing information about drivers, driver sets, and eDirectory. Long traces can be hard to read and confusing at times if multiple drivers are running on the server at the same time. This section focuses on explaining how Identity Manager tracing works.

By enabling trace, you can analyze the type of operation and data flowing between systems or the changes made by a driver's logic in both data and operations. For example, a trace can show you an event before entering a policy, so the starting conditions are known. You can see when a rule executes, evaluation of its conditions, whether the rule is selected, actions taken, and the tokens evaluated. The resulting document is shown after leaving the rules in the policy. You can configure the trace to print a text file containing messages about the status of Identity Manager processes for further analysis. This is helpful while developing drivers and for troubleshooting issues in a production environment.

NOTE: Use tracing only during testing and troubleshooting Identity Manager. Running it in a production environment increases the load on the Identity Manager server and can cause slow processing of the events.

Prerequisites

To correctly interpret the trace messages, you must have a good understanding of the following:

- Identity Manager core architecture. For more information, see What Are Policies? in the *NetIQ Identity Manager Understanding Policies Guide*.

- Flow of policies. For more information, see How Policies Function in the *NetIQ Identity Manager Understanding Policies Guide*.
- Identity Vault schema. For more information, see Managing the Schema (https://www.netiq.com/documentation/edirectory-9/edir_admin/data/a4a9bz0.html).
- DTD (Document Type Definition) for DirXMLScript and XDS (NDS DTD). For more information, see DTD Reference (https://www.netiq.com/documentation/identity-manager-developer/dtd-documentation.html).
  - **<input>** and **<output>** tags in NDS DTD.
    - **Input:** Encapsulates events or commands sent as input to a driver or Identity Manager. All `<nds>` documents sent as a parameter to Identity Manager or driver interface method should contain exactly one `<input>` tag.
    - **Output:** Result of the events or commands returned to a driver or Identity Manager. All `<nds>` documents returned from Identity Manager or the driver interface method should contain exactly one `<output>` tag.

Configuring Trace

You can configure trace on a driver set, driver, or the Remote Loader.

Depending on the trace level specified for a driver, trace displays driver-related events when the engine processes the events. The driver trace level affects only the driver or driver set where trace is set. If you are using the Remote Loader, the Remote Loader trace file is set directly on the Remote Loader and contains only the driver shim trace.

Configuring Trace Levels

You can configure trace levels in Designer or Identity Console.

If you are using Remote Loader with a driver, configure the driver shim trace in the Remote Loader console. For more information, see Trace File Settings in "Understanding the Configuration Parameters for the Remote Loader" on page 34.

The following table describes the trace settings:

Table C-1 *Trace Settings*

Parameter	Driver	Driver Set
Trace level	As the driver trace level increases, the amount of information displayed in Trace increases. Trace level one shows errors, but not the cause of the errors. If you want to see password synchronization information, set the trace level to five. If you select **Use setting from Driver Set**, the value is taken from the driver set.	As the driver trace level increases, the amount of information displayed in Trace increases. Trace level one shows errors, but not the cause of the errors. If you want to see password synchronization information, set the trace level to five.

Parameter	Driver	Driver Set
Trace file	Specify a file name and location of 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.	Not Applicable
XSL trace level	Not Applicable	Trace 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	Not Applicable	Allows developers to attach a Java debugger. Restart the Identity Vault after attaching the Java debugger.
Trace file encoding	The trace file uses the system's default encoding. You can specify another encoding if desired.	Not Applicable
Java trace file	Not Applicable	When a value is set in this field, all Java information for the driver set is written to a file. The value for this field is the patch 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:** If the file size limit is specified 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. If you select **Use setting from Driver Set**, the value is taken from the driver set.	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:** If the file size limit is specified 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 if the driver name is very long.	Not Applicable

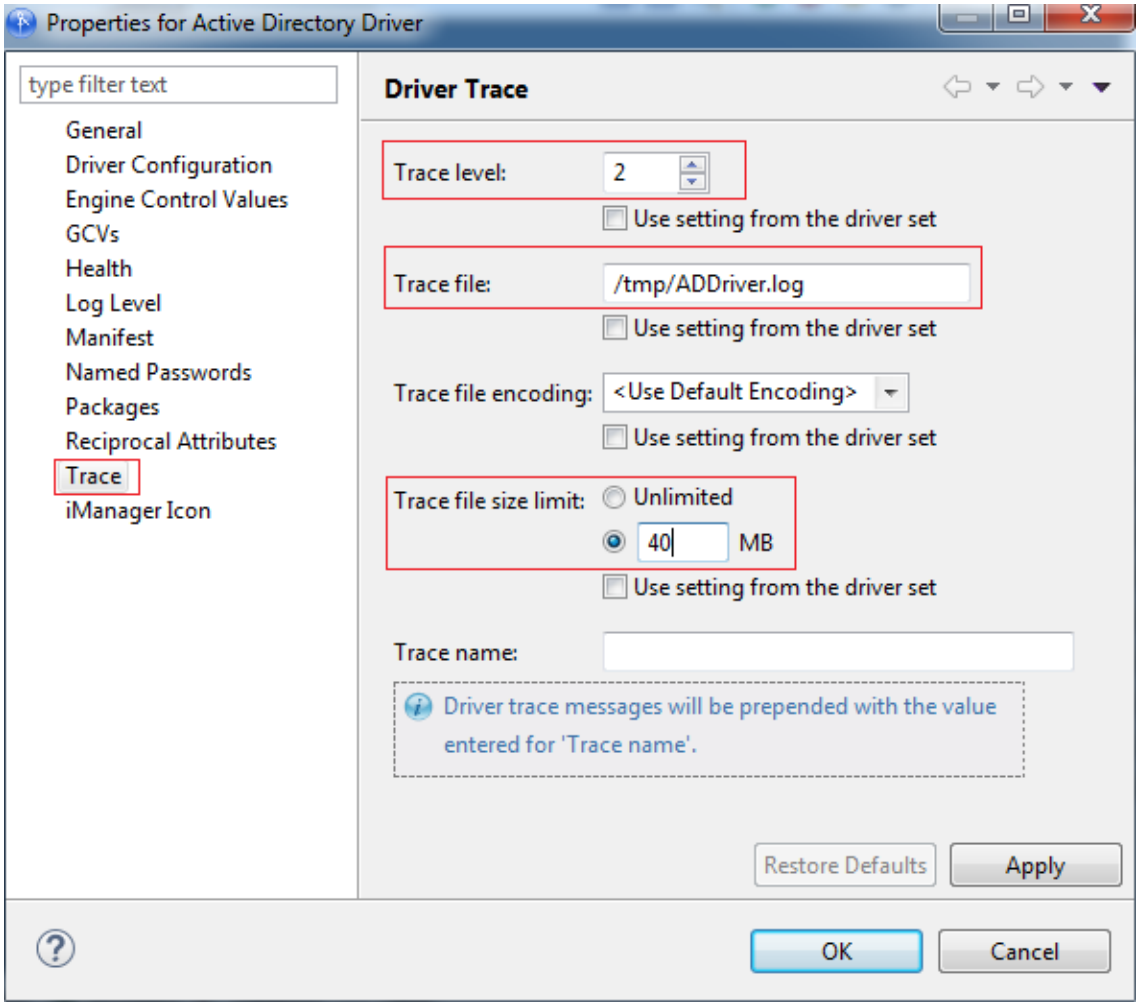
Configuring Trace Settings in Designer

You can add trace levels to each driver or to the driver set:

Driver Trace

In an open project in Designer, select the driver in the Outline view.

1. Open your project in Designer.
2. Select the driver in the Outline view.
3. Right-click and select **Properties**, then click **Trace**.
4. Set the parameters for tracing, then click **OK**.



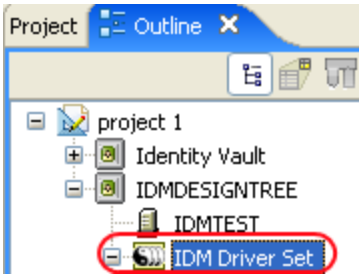
See Table C-1 for more information about the driver trace parameters.

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

Driver Set Trace

Setting the trace level on the driver set creates very long traces that are hard to read. All events from all drivers are included in one trace file. If you are troubleshooting an issue, it is best to set the trace only on the driver you are troubleshooting.

1. Open your project in Designer.
2. Select the driver set in the Outline view.
3. Right-click and select **Properties**, then click **Trace**.
4. Set the parameters for tracing, then click **OK**.



See Table C-1 for more information about the driver set trace parameters.

If you set the trace level on the driver set, all drivers appear in the trace logs.

Configuring Trace Settings in Identity Console

You can set the trace levels to each driver or to the driver set.

Driver Trace

1. Click the **IDM Administration** tab from the Identity Console home screen.
2. Click the specific driver set object on the right hand side of your computer screen to display all the drivers associated to it.
3. Select the specific driver object.
4. Click **Log & Trace Configuration** > **Trace** tab.
5. Set the parameters for tracing, then click **Save**.

   For information about these parameters, see Table C-1.

Driver Set Trace

1. Click the **IDM Administration** tab from the Identity Console home screen.
2. Click the Action menu icon of the appropriate **Driver Set > Driver Set Properties > Log & Trace Configuration > Trace**.
3. Set the parameters for tracing, then click **Save**.

   For information about these parameters, see Table C-1.

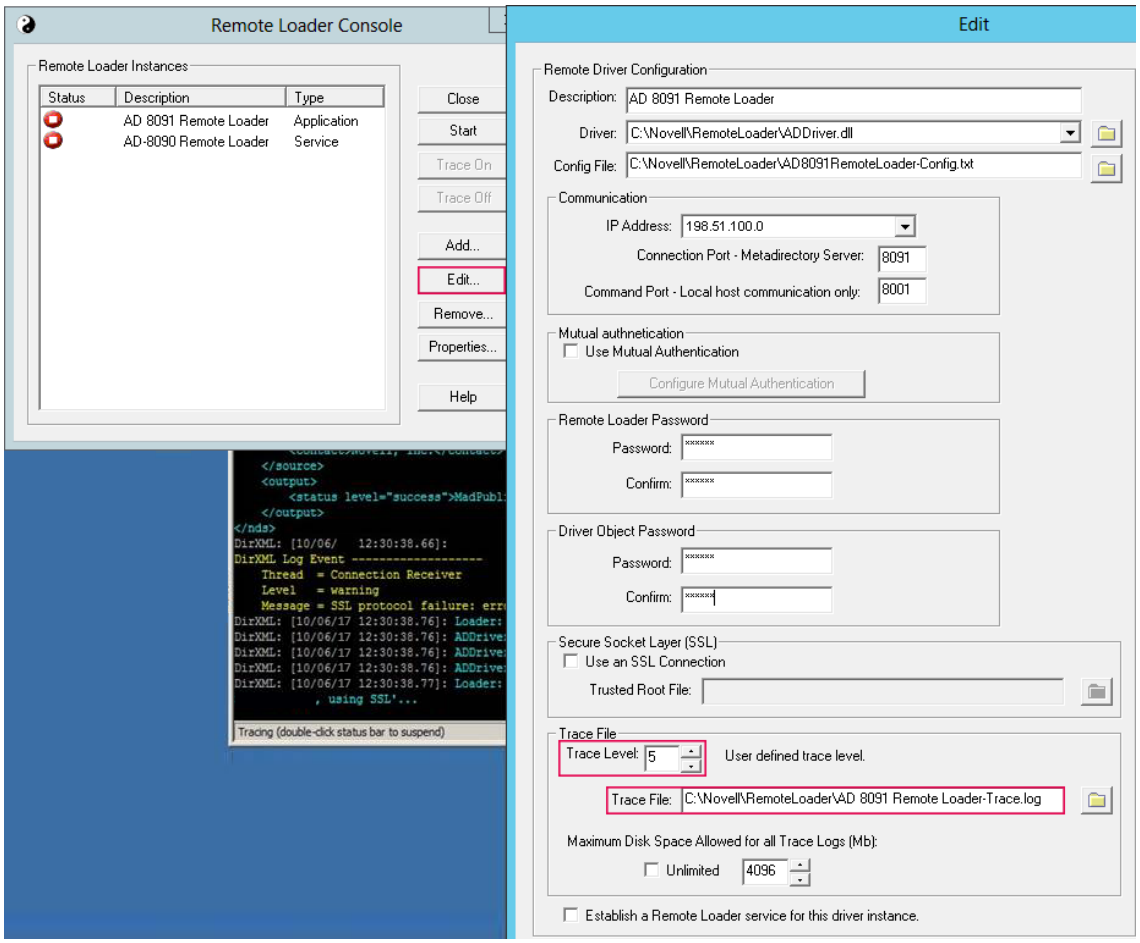
Configuring a Remote Loader Trace

You can capture the events that occur on the computer running the Remote Loader service by using one of the following methods:

Remote Loader Console

1 Launch the Remote Loader console by clicking the **Remote Loader Console** icon.

2 Select the driver instance, then click **Edit**.



3 Provide a value for **Trace Level**.

4 Specify a location and file for the trace file.

5 Specify the amount of disk space that the file is allowed.

6 Click **OK** twice to save the changes.

Command Line

You can enable tracing from the command line by using the switches in the following table:

Table C-2 Command Line Tracing Switches

Switch	Secondary Name	Parameter	Description
-trace	-t	integer	Specifies the trace level. This is used only when hosting an application shim. Trace levels correspond to those used on the Identity Manager server. Example: `-trace 3` or `-t3`

-tracefile	-tf	filename	Specifies a file to write trace messages to. Trace messages are written to the file if the trace level is greater than zero. Trace messages are written to the file even if the trace window is not open.
			
			Example:
			
			`-tracefile c:\temp\trace.txt`
			
			or
			
			`-tf c:\temp\trace.txt`

-tracefilemax	-tfm	size	Specifies the approximate maximum size that trace file data can occupy on disk. If you specify this option, there is a trace file with the name specified using the tracefile option and up to 9 additional "roll-over" files. The roll-over files are named using the base of the main trace filename plus "_n", where n is 1 through 9. The size parameter is the number of bytes. Specify the size by using the suffixes K, M, or G for kilobytes, megabytes, or gigabytes. If the trace file data is larger than the specified maximum when the Remote Loader is started, the trace file data remains larger than the specified maximum until roll-over is completed through all 10 files. Example: `-tracefilemax 1000M or -tfm 1000M`

Understanding the Trace Messages

The trace displays status and checkpoint messages about Identity Manager processes, XDS documents, DirXML Script, and engine processing.

- Status Messages
- Sample Success Status Message
- Sample Error Status Message
- Checkpoint Messages
- Policy Execution Messages
- Sample Trace Settings for Drivers

Status Messages

These messages contain different information depending on the level the trace is set and where it is set. For example, you can trace the Identity Manager engine for data caching, read and write events to the Identity Vault, or logic processing. On the driver side, you can trace the driver shim communication with the connected system.

The engine trace levels range from 0 to 4. Each trace level shows all the status messages of the previous levels.

Engine Trace Level	Information Provided
0	Status messages only
1	Current location in the driver logic and status messages
2	Events in XML format, current location in the driver logic and status messages
3	Driver logic execution details and the DirXML Script, events in XML format, current location in the driver logic and status messages
4	Cache-related information about the event from the Identity Vault, driver logic execution details and the DirXML Script, events in XML format, current location in the driver logic and status messages

An Identity Manager engine trace set at level 3 and above shows all the steps performed by the engine while processing an event. It contains four basic types of messages.

Status	Description
success	Operation or event was successful
warning	Operation or event was partially successful
error	Operation or event failed
fatal	A fatal error occurred. The driver should be shut down
retry	Application server was unavailable. Send this event or operation later

Below are some sample status messages with a break-down of their structure:

Sample Success Status Message

```
[12/05/17 07:12:03.609]: SampleDriver ST:
DirXML Log Event -------------------
Driver: \LAB-TREE\Identity Managerorg\Driverset\SampleDriver
Channel: Subscriber
Object: \LAB-TREE\Identity Managerorg\users\sampleuser
Status: Success
```

- The first line contains timestamp, the driver's name, and the channel of the driver. ST specifies Subscriber channel.
- The second line: "DirXML Log Event ————————-" is standard and can be used to search for status messages in a trace.

- The third line lists the driver's object location in the Identity Vault, using slash format for the dn of the driver.
- The fourth line indicates the channel on which the event occurred. It is Subscriber in this example.
- The fifth line indicates the object in the Identity Vault affected by the event, using slash format for the dn of the user.
- The sixth line has a message describing the error. In most cases, the message is accompanied with an error code and/or a Java stack.

Sample Error Status Message

```
[12/05/17 07:13:01.790]: SampleDriver SST:
DirXML Log Event -------------------
Driver: \LAB-TREE\idmorg\Driverset\SampleDriver
Channel: Subscriber
Status: Error
Message: Code(-9075) Shutting down because DirXML engine evaluation period
has expired
```

- The first line contains timestamp, the driver's name, and the channel of the driver. SST specifies Subscriber Service channel.
- The second line: "DirXML Log Event ————————-" is standard and can be used to search for status messages in a trace.
- The third line lists the driver's object location in the Identity Vault, using slash format for the dn of the driver.
- The fourth line indicates the channel on which the event occurred. It is Subscriber in this example.
- The fifth line indicates that an error has occurred.
- The sixth line contains a message describing the error. In most cases, the message is accompanied by an error code and may be followed by a Java stack.

For each processed event, Identity Manager generates a DirXML log event audit message in the trace. It sends the status messages to Publisher or Subscriber status log attributes and the auditing solution if the system is configured to send events to them. This message may appear before or after the XML in the trace.

Most of the engine troubleshooting such as driver startup and driver logic issues can be performed at level 3. The trace captured from a driver shim shows additional information. The driver shim trace levels range from 0 to 10, where the type and amount of information provided for the driver shim varies for each driver. For more information, see the specific driver implementation guide on the Identity Manager Drivers Documentation Website (https://www.netiq.com/documentation/identity-manager-49-drivers/).

For example, the default trace file on a connected Linux and UNIX system is located at `/usr/local/nxdrv/logs/trace.log`. A large amount of debugging information can be written to this file. The trace level setting in `/etc/nxdrv.conf` is used to control what is written to the file.

Checkpoint Messages

These are single line messages that indicate the stage of a process that is being executed. For example, a message that points where the engine is while executing the driver's logic or interacting with the Identity Vault. For example,

```
[12/05/17 07:12:03.790]: SampleDriver ST: Start transaction.
```

The following table lists some of the checkpoint messages and their meaning:

Message	Meaning
Start transaction.	The Subscriber channel reads an Identity Vault event from Identity Manager engine's event cache to process it.
No event transformation policies.	The driver's logic does not have any event transformation policies to be processed.
Applying event transformation policies.	The logic contained in the driver's event transformation policies will be executed after this message.
No object matching policies.	The driver's logic does not have any object matching policies to be processed.
Applying object matching policies.	The logic contained in the driver's object matching policies will be executed after this message.
No object creation policies.	The driver's logic does not have any object creation policies to be processed.
Applying object creation policies.	The logic contained in the driver's object creation policies will be executed after this message.
No object placement policies.	The driver's logic does not have any object placement policies to be processed.
Applying object placement policies.	The logic contained in the driver's object placement policies will be executed after this message.
No schema mapping policies to output.	The driver's logic does not have any schema mapping policies to be processed when sending information to the connected system.
Applying schema mapping policies to output.	The logic contained in the driver's schema mapping policies will be executed after this message.
No schema mapping policies to input.	The driver's logic does not have any schema mapping policies to be processed when receiving information from the connected system.
Applying schema mapping policies to input.	The logic contained in the driver's schema mapping policies will be executed after this message.
No command transformation policies.	The driver's logic does not have any command transformation policies to be processed.
Applying command transformation policies.	The logic contained in the driver's command transformation policies will be executed after this message.
No output transformation policies.	The driver's logic does not have any output transformation policies to be processed.
Applying output transformation policies.	The logic contained in the driver's output transformation policies will be executed after this message.

Message	Meaning
No input transformation policies.	The driver's logic does not have any input transformation policies to be processed.
Applying input transformation policies.	The logic contained in the driver's input transformation policies will be executed after this message.
Pumping XDS to the Identity Vault.	An event or query in XML format will be executed on the Identity Vault, and an XML document will be returned with either a status or a query result.
Stripping operation data from input document	Excludes the XML fragment that has <operation-data> as root from the XML document that is being sent to the driver shim.
Restoring operation data to output document	Reinserts the XML fragment that has <operation-data> as root in the response XML document from the driver shim. This XML fragment was saved before the XML document was sent to the driver shim.
End transaction.	The event initiated on the previous Start transaction is finished. Any status messages related to the event will be displayed before this message.
Receiving DOM document from application.	An event that occurred on the connected application is received on the Publisher channel.
Applying XSLT policy.	The StyleSheet (XSLT code) located at this point on the driver was executed on the event's XML document.
Filtering out notification-only attributes.	Removes the attributes flagged as Notify in the driver's filter for the channel being processed.
Fixing up association references.	Converts the values of the Identity Vault attributes with dn or path syntax (object references) to the name of the associated object on the connected system.
Resolving association references.	Converts the values of attributes with dn or path syntax (object references) from the associated connected system object to the equivalent Identity Vault object.
Applying publisher filter.	Allows the attributes or classes flagged as Synchronize on the Publisher channel filter to pass. If attributes or classes are flagged as Ignore, they are excluded from the current event. Additionally, attributes or classes that are not listed in the filter are excluded.

Policy Execution Messages

The trace displays messages that specify which operation is being evaluated against the driver's logic. It specifies the policy, rule name, and the results of the evaluated rule's conditions and actions. For example,

```
[12/05/17 07:54:02.250]: SampleDriver PT: Applying to add #1.
```

The following table lists some sample policy execution messages and their meaning:

Message	Meaning
Applying policy: %+C%14CEducation_Loan %-C.	The Identity Manager engine executes the logic contained in the policy named `Education_Loan`. The name of the policy is always within the delimiters `%+C%14C` and `%-C`, and matches the name of a policy in the driver's logic.
Applying to operation #1.	The policy logic is applied to the operation listed. In this message, you will see one of the following strings: ◆ add ◆ add-association ◆ check-object-password ◆ check-password ◆ delete ◆ get-named-password ◆ init-params ◆ instance ◆ modify ◆ modify-association ◆ modify-password ◆ move ◆ password ◆ query ◆ query-schema ◆ remove-association ◆ rename ◆ schema-def ◆ status ◆ sync The number after the # sign is the position of that event within the `<input>` or `<output>` document, with number 1 being the first position.
Evaluating selection criteria for rule 'Rule_Allow'.	Indicates that the conditions coded in the `Rule_Allow` rule are being evaluated. The name of the rule is contained within single quotes, and will match the name of a rule in your driver's logic.
(if-operation equal "add") = TRUE.	Specifies that the engine is evaluating if-operation equal "add" DirXML Script code. The value after equal to (=) specifies the outcome of evaluating the code, which can be TRUE or FALSE. This code is the result of the logic entered in the driver through Policy Builder (Identity Console or Designer). For more information about If operation, see If in *NetIQ Identity Manager - Using Designer to Create Policies*.
Rule rejected.	Specifies that the rule is rejected because the rule's conditions were not completely met. Therefore, this rule's actions will not be executed.

Message	Meaning
Rule selected.	Specifies that the rule's conditions are met, so the rule's actions will be executed.
Applying rule 'Rule_Allow'.	Specifies that the actions included in the `Rule_Allow` rule will be executed. The name of the rule will always be within single quotes, and will match the name of a rule in your driver's logic.
Action: do-strip-opattr("nspmDistributionPass word").	Indicates that the action after the colon sign (:) will be executed. To understand the details of this action, see *NetIQ Identity Manager - Using Designer to Create Policies*.

Below are sample Identity Manager engine trace messages logged at level 3 along with a brief explanation for each message.

Message	Meaning
[12/05/17 08:29:22.625]: Active Directory ST: Applying policy: %+C%14C'Transform NMAS attribute to password elements'%-C.	Specifies that the engine is processing a policy called Transform NMAS attribute to password elements.
[12/05/17 08:29:22.625]: Active Directory ST: Applying to modify #1.	Specifies that the engine is executing a modify object operation.
[12/05/17 08:29:22.625]: Active Directory ST: Evaluating selection criteria for rule 'Convert adds of the nspmDistributionPassword attribute to password elements'.	Specifies that the engine is processing `Convert adds of the nspmDistributionPassword attribute to password elements` rule.
[12/05/17 08:29:22.625]: Active Directory ST: (if-operation equal "add") = FALSE.	Specifies the condition contained in the rule has evaluated to FALSE.
[12/05/17 08:29:22.625]: Active Directory ST: Rule rejected.	Indicates that not all necessary conditions for the rule were met, so the rule will not be processed.
[12/05/17 08:29:22.625]: Active Directory ST: Evaluating selection criteria for rule 'Block modifies for failed password publish operations if reset password is false'.	Indicates that the rule being processed is Block modifies for failed password publish operations if reset password is false.
[12/05/17 08:29:22.640]: Active Directory ST: (if-global-variable 'reset-externalpassword-on-failure' equal "false") = FALSE.	Specifies the condition contained in the rule has evaluated to FALSE.
[12/05/17 08:29:22.640]: Active Directory ST: Rule rejected.	Indicates that not all necessary conditions for the rule were met, so the rule will not be processed.
[12/05/17 08:29:22.640]: Active Directory ST: Evaluating selection criteria for rule 'Convert modifies of a nspmDistributionPassword attribute to a modify password operation'.	Specifies that the rule being processed is `Convert modifies of a nspmDistributionPassword attribute to a modify password operation`.
[12/05/17 08:29:22.640]: Active Directory ST: (if-operation equal "modify") = TRUE.	Specifies the condition contained in the rule has evaluated to TRUE.
[12/05/17 08:29:22.640]: Active Directory ST: Rule selected.	Indicates that all necessary conditions for the rule were met, so the rule will be processed.

Message	Meaning
[12/05/17 08:29:22.640]: Active Directory ST: Applying rule 'Convert modifies of a nspmDistributionPassword attribute to a modify password operation'.	Specifies that the actions included in the `Convert modifies of a nspmDistributionPassword attribute to a modify password operation` rule will be executed.
[12/05/17 08:29:22.640]: Active Directory ST: Action: do-set-dest-password(token-xpath("modify-attr[@attr-name='nspmDistributionPassword']//add-value//value")).	Specifies the action that will be taken as part of executing the `Convert modifies of a nspmDistributionPassword attribute to a modify password operation` rule.
[12/05/17 08:29:22.656]: Active Directory ST: arg-string(token-xpath("modify-attr[@attrname='nspmDistributionPassword']//add-value//value"))	Specifies the action that will be taken as part of executing the `Convert modifies of a nspmDistributionPassword attribute to a modify password operation` rule.
[12/05/17 08:29:22.656]: Active Directory ST: token-xpath("modify-attr[@attrname='nspmDistributionPassword']//add-value//value")	Specifies the action that will be taken as part of executing the `Convert modifies of a nspmDistributionPassword attribute to a modify password operation` rule.
[12/05/17 08:29:22.656]: Active Directory ST: Token Value: "-- suppressed --".	Specifies the action that will be taken as part of executing the `Convert modifies of a nspmDistributionPassword attribute to a modify password operation` rule.
[12/05/17 08:29:22.656]: Active Directory ST: Arg Value: "-- suppressed --".	Specifies the action that will be taken as part of executing the `Convert modifies of a nspmDistributionPassword attribute to a modify password operation` rule.
[12/05/17 08:29:22.656]: Active Directory ST: Action: do-strip-opattr("nspmDistributionPassword").	Specifies the action that will be taken as part of executing the `Convert modifies of a nspmDistributionPassword attribute to a modify password operation` rule.
[12/05/17 08:29:22.656]: Active Directory ST: Action: do-set-xml-attr("eventid","../modify-password","pwd-subscribe").	Specifies the action that will be taken as part of executing the `Convert modifies of a nspmDistributionPassword attribute to a modify password operation` rule.
[12/05/17 08:29:22.656]: Active Directory ST: arg-string("pwd-subscribe")	Specifies the action that will be taken as part of executing the `Convert modifies of a nspmDistributionPassword attribute to a modify password operation` rule.
[12/05/17 08:29:22.656]: Active Directory ST: token-text("pwd-subscribe")	Specifies the action that will be taken as part of executing the `Convert modifies of a nspmDistributionPassword attribute to a modify password operation` rule.
[12/05/17 08:29:22.671]: Active Directory ST: Arg Value: "pwd-subscribe".	Specifies the action that will be taken as part of executing the `Convert modifies of a nspmDistributionPassword attribute to a modify password operation` rule.

Sample Trace Settings for Drivers

The following table describes sample driver trace settings and the information logged with those settings:

Identity Manager Driver	Engine Trace Level	Shim Trace Level	Description
Active Directory	3	3	◆ Helps to troubleshoot when users do not synchronize. Note the username, location, and the system from the trace. ◆ Helps to troubleshoot when users synchronize in a single direction. You must perform the following actions to resolve the issue: ◆ Check the driver filters ◆ Check the placement policies in the appropriate channel Shim trace level 5 shows details about password synchronization.
Delimited Text	3	3	◆ Helps to troubleshoot when users are not created in the Identity Vault. You must perform the following actions to resolve the issue: ◆ Check if the input directory exists and is properly entered in the driver configuration ◆ Check the filesystem rights and quotas on input directory and files ◆ Input CSV file is used to create the users ◆ Helps to troubleshoot when the driver does not write output files. You must perform the following actions to resolve the issue: ◆ Check if the output directory exists and is properly entered in the driver configuration ◆ Check the file system rights and quotas on output directory
eDirectory	3		The eDirectory driver works in pairs. The driver does not support Remote Loader. To troubleshoot any issues, ensure the following prerequisites are met: ◆ For the driver exports, ensure that you get both eDirectory driver exports. ◆ Ensure that both eDirectory drivers are imported in your Designer project.

Identity Manager Driver	Engine Trace Level	Shim Trace Level	Description
Entitlements Service	5		This driver enables or disables entitlements on objects. To troubleshoot any issues, ensure that the following prerequisites are met: ◆ LDAP export of the entitlement policies is used in the driver set containing the driver ◆ This driver changes only the DirXML-EntitlementRef attribute on a user. Ensure that you obtain appropriate traces on the other drivers being affected by this change.
GroupWise	3	5	Helps to troubleshoot any issues when mail accounts are not created in GroupWise.
ID Provider			This is a service driver with which external clients can be accessed. ◆ Traces can be enabled in the driver and client parameters other than the regular Identity Manager tracing. ◆ The following information helps to analyze the cause of error and troubleshoot the driver: ◆ Document the issue in detail ◆ Obtain the ID driver export ◆ Get the LDAP export of the ID policy objects ◆ Obtain the XSLT / Java call made to the ID Provider service
JDBC	3	3	The following information helps to analyze the cause of error and troubleshoot the driver: ◆ Database name, vendor and patch level ◆ Operating System and patch level where the database is running ◆ Whether it is a supported Identity Manager/ database combination ◆ Driver connection mode ◆ Schema of the target database A shim trace level of 5 shows information about SQL commands that are executed by the driver shim. Additional debugging information is shown with level 7.
JMS	3	3	Helps to troubleshoot any issues when messages are not sent or received from the JMS queue or application.
LDAP	3	5	Helps to troubleshoot any issue when users are not synchronizing between systems.

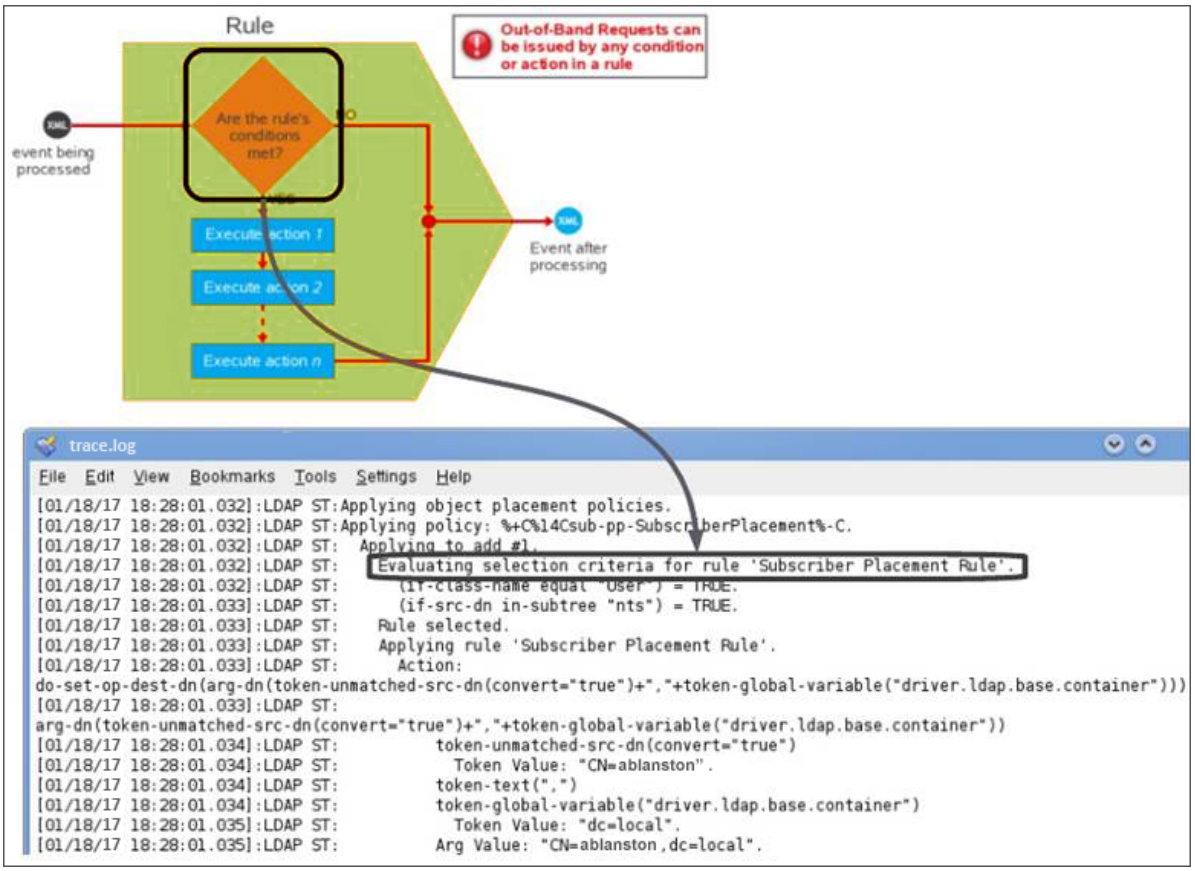
Identity Manager Driver	Engine Trace Level	Shim Trace Level	Description
Linux and UNIX Settings	10		Helps to troubleshoot any issue when attributes are not added to the newly created users.
Linux and UNIX Bi-directional	3	4	Helps to troubleshoot any issues when a user is not created on the platform, or data is not correctly synchronizing after the user is created.
Loopback	3		Helps to troubleshoot any issues.
Lotus Notes	3	5	Helps to troubleshoot any issue.
Manual Task Service	5		Helps to troubleshoot any issue.
PeopleSoft 5.2	3	5	Helps to troubleshoot any issue other than the connectivity issue.
Salesforce	3	5	Helps to troubleshoot any issues.
SAP HR	3	5	Helps to troubleshoot issues when objects are not synchronized to SAP.
SAP User Management	3	5	Helps to troubleshoot any issue other than the connectivity issue.
SOAP	3	5	Helps to troubleshoot connectivity issues with the SOAP system. Engine trace level 3 helps to troubleshoot any issues.
WorkOrder	3	5	Helps to troubleshoot any issues.

Interpreting Trace During Identity Manager Operations

- When a Rule Executes
- During Driver Startup
- During Query Processing
- When Direct Commands are Processed
- During add-association Events
- Cases with No Trace Excerpts

When a Rule Executes

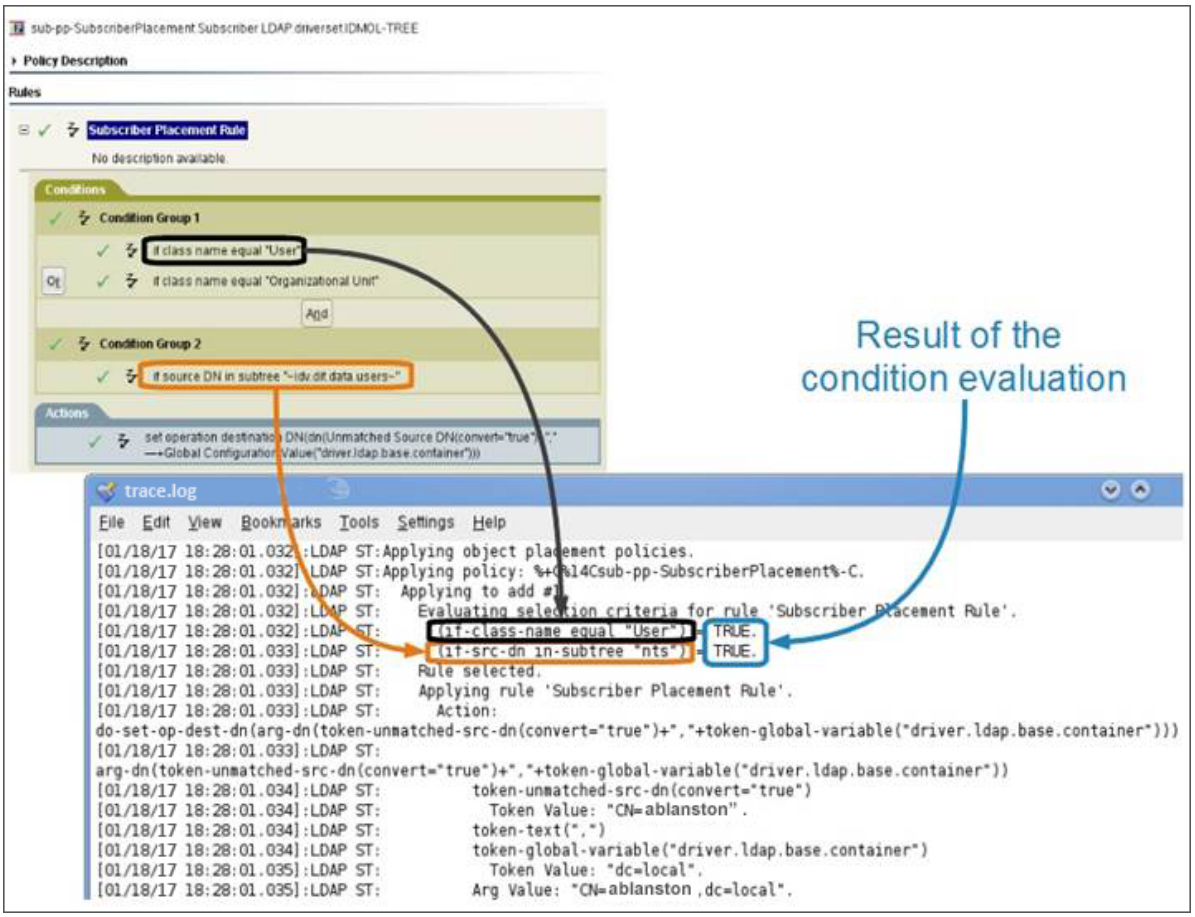
An Identity Manager policy contains one or more rules. Each rule comprises of one or more conditions and actions that will be performed when the conditions are met. When Identity Manager engine evaluates the condition set for a rule, the trace displays `Evaluating Selection Criteria` for the rule followed by the rule's name.



If the complete condition is true, the rule execute its actions. If the condition is false, the rule's actions are skipped and the rule is rejected.

The rule rejected message does not mean that there is an issue in the logic. Each rule reacts to certain events and conditions. It is normal that several rules are skipped. For example, you instructed Identity Manager to reject rules that handle only users when processing a group event while synchronizing both users and groups.

Before a rule executes its actions, the trace displays `Applying Rule` followed by the rule's name. Identity Manager evaluates the conditions using boolean logic. Note that the trace displays a condition group in parenthesis and the outcome of evaluating the logic of the condition group after the parenthesis. This information is useful particularly when the condition is comparing something to true or false.



The trace shows each action in the rule's logic and also displays details about the processing needed to evaluate the arguments of the action. The first line contains the actual action that will be executed and its arguments. This is followed by each individual portion of the argument being resolved. It then shows the argument's value that will be used in the action.

In this example, the argument for Set Operation Destination DN command is composed of three items: an unmatched Source DN token, a comma character, and the value of a global variable. The final value of the argument (`cn=ablanston,dc=local`) that Identity Manager uses in the command appears after Arg Value.

During Driver Startup

You can capture useful information about a driver in a trace, such as GCV information, initialization parameters, and filter information during the driver initialization.

When the driver initialization process starts, the driver reads the named passwords list, information from the driver object, and the objects that store the actual policies, rules, and stylesheets. During this process, the trace displays messages such as Reading XML attribute vnd.nds.stream; full DN of the object, and the attribute that the driver reads in addition to other information that the driver reads.

Below is a sample trace from an eDirectory driver startup:

```
[01/02/17 10:26:15.451]:eDirectory Driver :Reading named passwords list.
[01/02/17 10:26:15.469]:eDirectory Driver :Named passwords:
[01/02/17 10:26:15.570]:eDirectory Driver :Reading XML attribute
vnd.nds.stream://OESNW_TREE/idm/services/driverset/
eDirectory+Driver#DirXML-EngineControlValues.
[01/02/17 10:26:16.113]:eDirectory Driver :Reading XML attribute
vnd.nds.stream://OESNW_TREE/idm/services/driverset#DirXML-ConfigValues.
[01/02/17 10:26:16.126]:eDirectory Driver :Reading XML attribute
vnd.nds.stream://OESNW_TREE/idm/services/driverset/
eDirectory+Driver#DirXML-ConfigValues.
[01/02/17 10:26:16.180]:eDirectory Driver :Global Configuration Values:
[01/02/17 10:26:16.182]:eDirectory Driver : Name: enable-password-
subscribe Value: true
[01/02/17 10:26:16.184]:eDirectory Driver : Name: enable-password-publish
Value: true
[01/02/17 10:26:16.185]:eDirectory Driver : Name: publish-password-to-nds
Value: true
[01/02/17 10:26:16.186]:eDirectory Driver : Name: publish-password-to-dp
Value: false
[01/02/17 10:26:16.188]:eDirectory Driver : Name: enforce-password-policy
Value: true
[01/02/17 10:26:16.190]:eDirectory Driver : Name: reset-external-password-
on-failure Value:
true
[01/02/17 10:26:16.205]:eDirectory Driver : Name: notify-user-on-password-
dist-failure Value:
true
[01/02/17 10:26:16.206]:eDirectory Driver : Name: ConnectedSystemName
Value: eDirectory Driver
[01/02/17 10:26:16.208]:eDirectory Driver : Name: dirxml.auto.treename
Value: OESNW_TREE
[01/02/17 10:26:16.209]:eDirectory Driver : Name: dirxml.auto.driverdn
Value: \OESNW_TREE\idm\services\driverset\eDirectory Driver
[01/02/17 10:26:16.223]:eDirectory Driver : Name: dirxml.auto.driverguid
Value: {044F4400-
B953-11dc-968B-000C290066AE}
```

Below is the meaning of each line in the trace:

1. The first step is the driver start. At this point, the driver tries to read the named passwords list.
2. There is a header before listing the names on the named password list. The header is blank because this driver did not have any named passwords configured.
3. The engine reads the value from the DirXML-EngineControlValues attribute. This attribute resides in the Identity Vault object called Identity Vault Driver under the driverset.services.Identity Manager container under the tree named OESNW_TREE.
4. The engine reads the value from the DirXML-ConfigValues attribute that resides in the Identity Vault object called driverset under services.Identity Manager container under the tree named OESNW_TREE.
5. The engine reads the value from the DirXML-ConfigValues attribute that resides in the Identity Vault object called Identity Vault Driver under driverset.services.Identity Manager container under the tree named OESNW_TREE.
6. A header is shown before listing the Global Configuration Values.

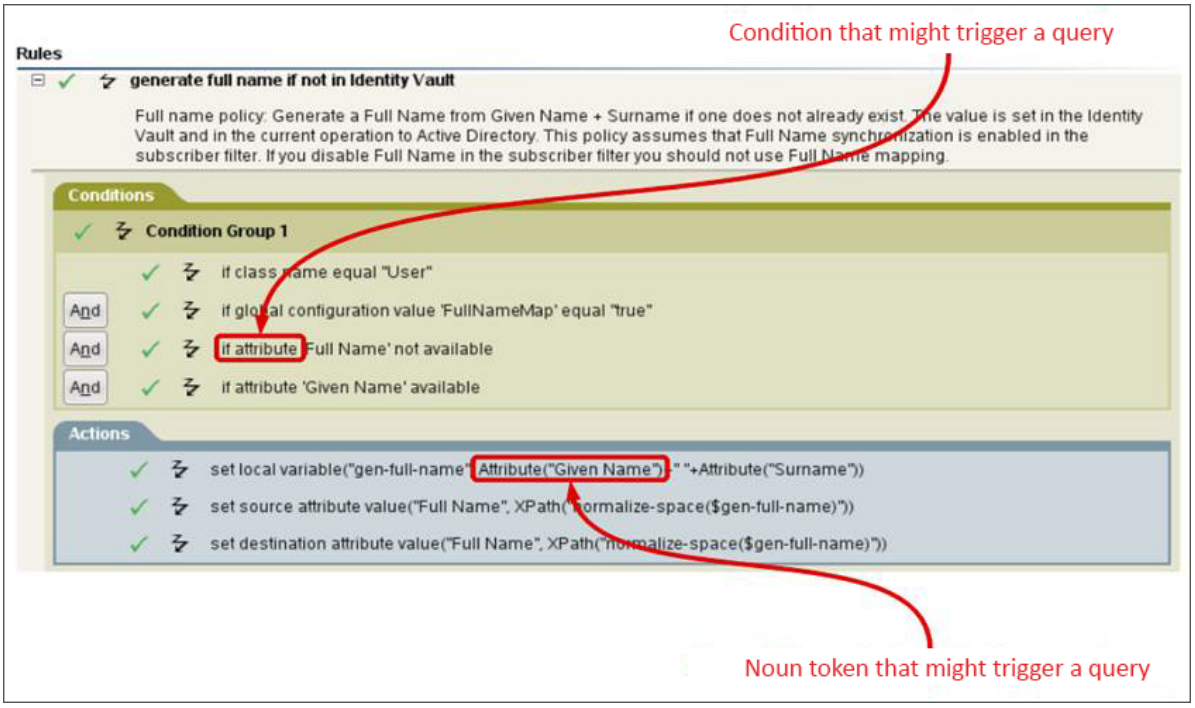
7. This and the remaining lines show the GCV's names and values that will be used in the policy's logic.
8. As the driver initialization continues, the driver reads the values for certain attributes and shows them in the trace. If the driver is using the Remote Loader, you also need to see the Remote Loader connection messages in the Remote Loader trace. The driver initialization continues reading information from the needed objects until it finishes initializing all of them and then proceeds to query the shim for identifying the driver class.
9. The shim responds to the query with information about its type, version, activation level, and capabilities. There is a possibility that some queries are sent and messages are exchanged in the process, such as a query from the shim to the driver for its GUID and Public Key attributes. The driver initialization completes when you see the following message in the trace:

   ```
   [12/06/17 10:26:26.467]: Identity Vault Driver ST:Transitioned from
   state '%+C%14CStarting%-C' to state '%+C%14CRunning%-C'.
   ```

10. After this message, the events start coming in the trace. If you are looking at fixing the driver initialization problems, set the engine trace to level 4 and read the information line by line until you see an error message with more details.

During Query Processing

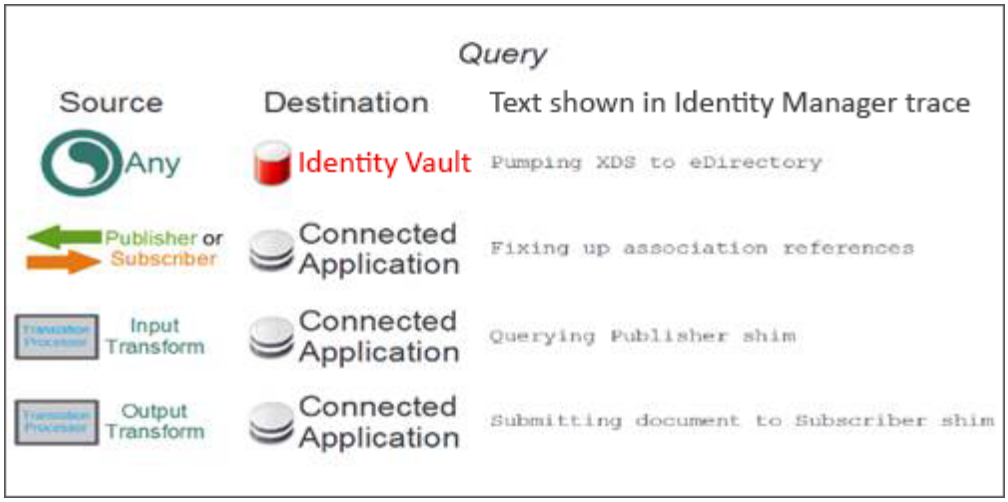
A query is an out of band request that stops the normal flow of events until it is processed. The engine stops processing the current event until the query is processed. A query can be issued by a condition within a rule, by an action within a rule, or by certain noun tokens inside an action's parameters. You can issue a query within XSLT code. Such queries act exactly the same way as the query issued by using DirXML Script.



The path of a query path varies based on its source and destination. Therefore, information printed in the trace also varies.

A query can follow the following paths:

- Straight into Identity Vault without passing through policies.
- To the application, but passing the channel-independent translation logical block.
- Straight into the application without passing through policies.

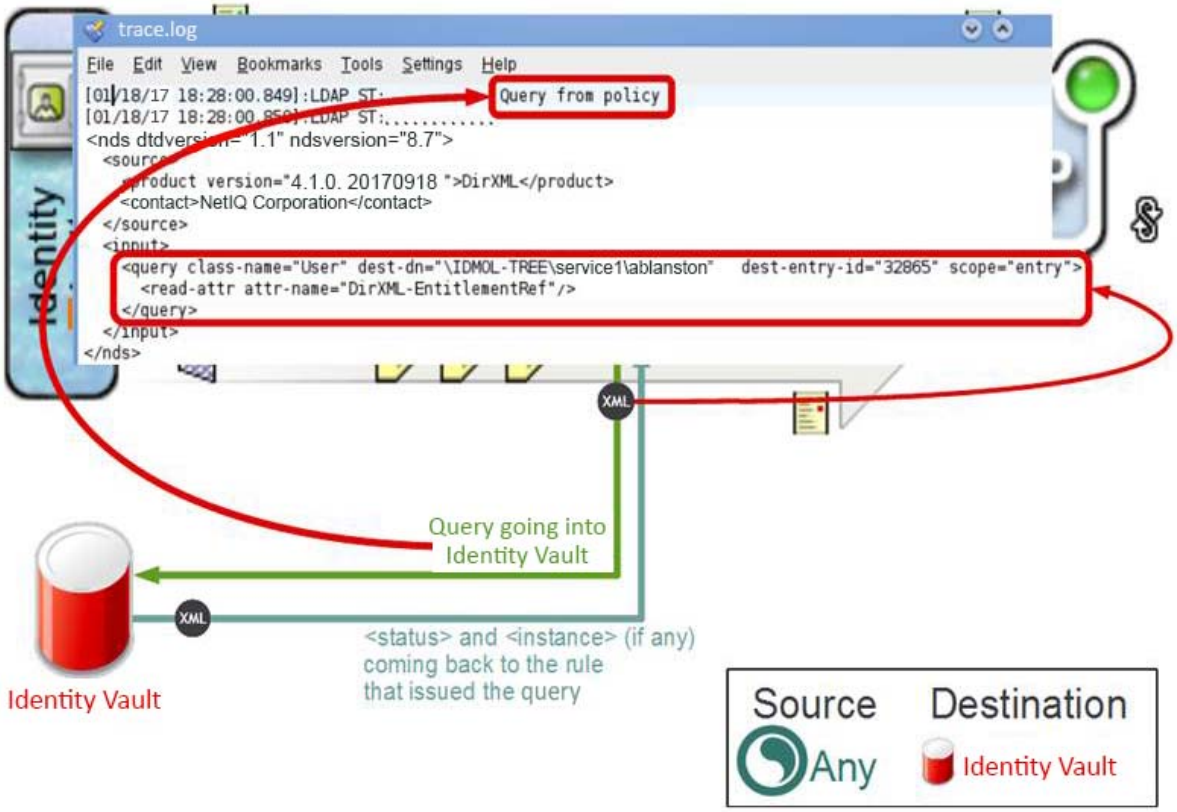


Example 1

A query from a channel is sent to the connected application and passes through the translation logic. At the same time, policies and rules in the driver's Subscriber channel are processing a modify event from the Identity Vault.

When a rule issues a query, the original modify event stops processing until the query results are returned. The result of a query always returns a status message. A success in the status message means that the query is issued without errors. However, it does not mean that any results are returned.

A query in the Identity Manager trace (level 2 or above) prints `"Query from policy"` text. The below figure shows the query element containing `User` class with single entry as the scope of the search for the value of DirXML-EntitlementRef attribute.



The trace prints `Pumping XDS to eDirectory` message when the query enters the Identity Vault. This message is printed every time an event or a command enters the Identity Vault. When the result of the query is received, the trace prints `Query from policy result` message. The result always contains a status element and an instance element with a single object. In case no objects are found, the Identity Vault returns no instance elements. It returns one instance element for each object.

After the results of the query are returned to the rule that issued it, the original event processing is resumed.

Example 2

A query passes through the Outbound Association Reference Processor, Schema Mapping policy set, and Output Transformation policy set of the engine, and then handed over to the driver shim before it is passed to the connected application.

When a rule (one from the Event Transformation Policy set) issues a query, the original modify event stops processing until the query results are returned.

The connected application returns the results of the query to the driver shim. The driver shim then builds the XDS document with a status element and zero or more instance elements. This XDS document then passes through the Input Transformation policy set, Schema Mapping policy set, and Inbound Association Reference Processor before it returns to the rule that issued the original query.

Note that this is the only case where both query and its elements pass through the driver logic. This occurs because Identity Manager assumes that anything issued from within a driver channel is in the Identity Vault format. For the application to understand what is requested, the event must converted to the application format. Conversely, any results from the application must converted back to the Identity Vault format before it is processed.

After the results of the query are returned to the rule that issued it, the original event resumes processing.

Example 3

A query passes through the Outbound Association Reference Processor, Schema Mapping policy set, and Output Transformation policy set of the engine, before it is passed to the connected application.

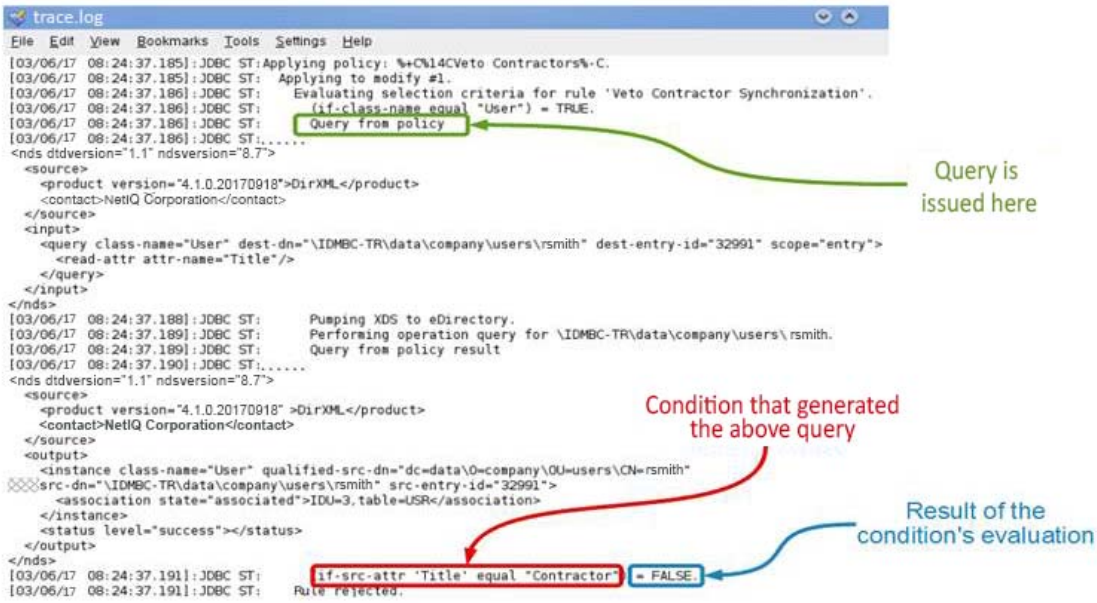
When a rule (one from the Output Transformation Policy set) issues a query, the original modify event stops processing until the query results are returned. In this case, the query is directly passed to the driver shim. The driver shim directly returns the query result to the rule that issued the query.

After sending the query result, the driver resumes processing the original event.

Identity Manager processes queries in the same way regardless of whether the query is issued from a condition or an action. However, the trace displays changes based on the source that issued the query.

When a query is issued from a condition

The condition that issued the query appears only after the query result is returned. This is because the trace shows a rule's condition with its evaluation result on the same line. To evaluate this condition, Identity Manager engine needs the query's results.



When a query is issued from within an action

The action is shown before the query is issued. This is because the result is not necessary to display the action in the trace.

```
trace.log
File  Edit  View  Bookmarks  Tools  Settings  Help
[01/18/17 18:28:00.848]:LDAP ST:    Applying rule 'Match Users on Surname and Given Name'.
[01/18/17 18:28:00.848]:LDAP ST:      Action: do-implement-entitlement(arg-node-set(token-entitlement("Acco
[01/18/17 18:28:00.849]:LDAP ST:        arg-node-set(token-entitlement("Account"))
[01/18/17 18:28:00.849]:LDAP ST:          token-entitlement("Account")
[01/18/17 18:28:00.849]:LDAP ST:            Query from policy
[01/18/17 18:28:00.850]:LDAP ST:............
<nds dtdversion="1.1" ndsversion="8.7">
  <source>
    <product version="4.1.0. 20170918 ">DirXML</product>
    <contact>NetIQ Corporation</contact>
  </source>
  <input>
    <query class-name="User" dest-dn="\IDMOL-TREE\service1\ablanston" dest-entry-id="32865" scope="entry">
      <read-attr attr-name="DirXML-EntitlementRef"/>
    </query>
  </input>
</nds>
[01/18/17 18:28:00.850]:LDAP ST:            Pumping XDS to eDirectory.
[01/18/17 18:28:00.851]:LDAP ST:            Performing operation query for \IDMOL-TREE\service1\ablanston.
[01/18/17 18:28:00.852]:LDAP ST:            Query from policy result
[01/18/17 18:28:00.852]:LDAP ST:............
<nds dtdversion="1.1" ndsversion="8.7">
  <source>
    <product version="4.1.0. 20170918">DirXML</product>
    <contact>NetIQ Corporation</contact>
  </source>
  <output>
    <instance class-name="User" qualified-src-dn="O=service1\CN=ablanston" src-dn="\IDMOL-TREE\service1\ablanston"
src-entry-id="32865"/>
    <status level="success"></status>
  </output>
</nds>
[01/18/17 18:28:00.862]:LDAP ST:          Token Value: {}.
[01/18/17 18:28:00.862]:LDAP ST:          Arg Value: {}.
```

Query is issued here

Condition that generates the query

When Direct Commands are Processed

Direct commands can follow three paths just like queries.

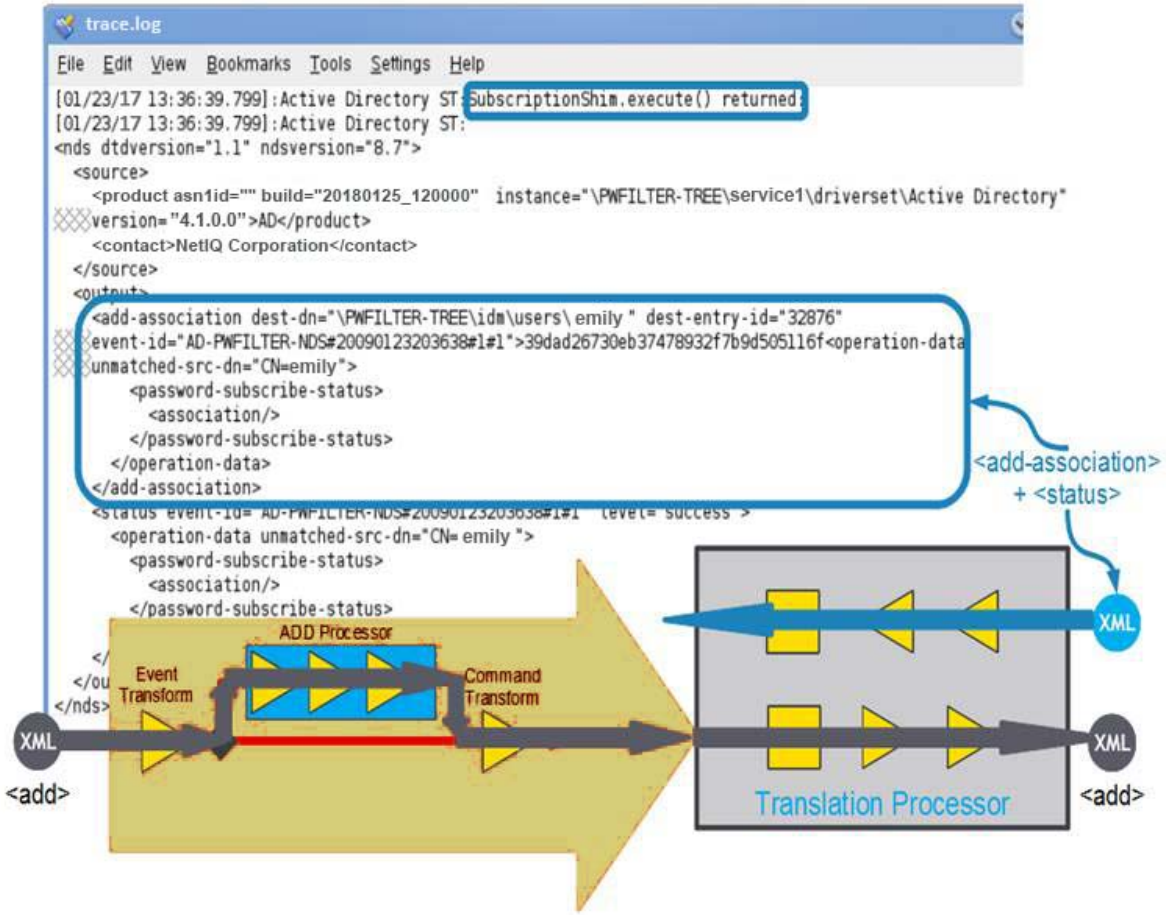
- Straight into Identity Vault without passing any policies
- To the application, but passing the channel-independent translation logic
- Straight into the application, without passing policies

The trace shows a command preceded by `Direct command from policy` text. It is easy to track the destination and path of a command by reading the first line in the trace after the command is issued.

During add-association Events

When Identity Vault sends an add event to a driver shim, and if the driver shim executes the event in the connected application, it returns both the status message and an add-association command to the Identity Vault. This command follows the same flow as the status message before it is processed.

The below example shows the XDS for an <add-association> event. The driver sends `39dad26730eb37478932f7b9d505116f` as an association value to the Identity Vault. This is a value that the shim knows that uniquely identifies the newly created entry in the connected application.



The <add-association> ties back to the original event through its event-id XML attribute. The driver shim also generates the destination DN by copying the source-DN of the original event.

In addition, the driver shim sends some additional XML elements (tags) along with the association value, but they are not part of it. The status message that the driver sends also contains those XML elements. Those XML elements are the properties of the operation. Operation properties are never sent to the shim. The Identity Manager engine strips the properties from the source event before handing over the event to the driver shim, and adds them back to the result of the event.

Since the properties of the operation are not sent to the shim and are added back to the returned results, the Identity Manager engine uses these properties to save information about the original event. This information is used in the logic that processes those results.

The add-association event then passes the translation processor, and right after that the destination object in the Identity Vault obtains an association. An add event from the Publisher channel already has the association value in the event that the engine uses to build the association when it creates the object in the Identity Vault. Therefore, you do not see an add-association command.

The add association rule can forcefully add an association to an object in the Identity Vault. For more information about the rule, see Add Association in the *NetIQ Identity Manager - Using Designer to Create Policies*.

Cases with No Trace Excerpts

There are certain cases where trace is not printed. For example, policies and rules in the driver's Subscriber channel are processing a modify event from the Identity Vault and a command is sent from inside a channel to the connected application.

When a rule issues a direct command, the original modify event stops processing until a status message is received from the command. In this case, the modify event issued by the rule passes through the data transformation logical blocks of the engine, the driver shim, and then reaches the connected application. The connected application then performs the required action and returns a status message (success or error status). The driver shim builds the XDS document with a status element that contains the result. This XDS document passes through the data transformation logical blocks before it is returned to the rule that issued the command.

Note that this is the only case where both modify event and its resulting status elements pass through the driver logic. This occurs because Identity Manager assumes that anything issued from within a driver channel is in Identity Vault format. For the application to understand what is being requested, the event must be converted to the application format. Conversely, any result from the application must be converted to an Identity Vault format before it can be processed.

When the rule receives the result of its direct command, Identity Manager resumes processing the original event.

Example

Policies and rules in the driver's Subscriber channel are processing a modify event from the Identity Vault and a policy inside the Input Transformation policy set or the Output Transformation policy set issues a direct command to the application.

When a rule issues a command, the original modify event stops processing until the command's results are returned. In this case, the modify event is directly handed to the driver shim. The driver shim directly sends the result of the event to the rule that issued it.

When the results of the command are returned to the rule that issued it, the driver resumes processing the original event.

Both queries and direct commands follow very similar paths in the engine, and these paths are different from the regular event flow. A driver can have several instances of queries and direct commands. It is a good practice to remember the information that the trace displays for them when they start and stop.

D The Cache Flush Parameter

Identity Manager provides an option to turn off the file system flush for each write. If you disable cache writes, they are not flushed immediately. Instead, it is left to the underlying operating system to take care of file system writes. Though this approach improves the performance on systems where there are heavy writes, NetIQ recommends that you do not use this approach for production systems.

NOTE: Turning off the file system flush is supported only on Linux.

To turn off the file system flush, set the environment variable `DIRXML_SKIP_FSYNC` to some value and restart eDirectory. The code only looks for the presence of this environment variable and does not regard the value associated with it. Alternatively, you can set the environment variable in the pre-ndsd start script, as shown below:

```
DIRXML_SKIP_FSYNC=true # Added manually to skip file system flush

export DIRXML_SKIP_FSYNC # Added manually to skip file system flush
```

To turn on the file system flush, remove the environment variable setting and then restart eDirectory. If you have set the environment variables in the pre-ndsd script, remove them and then restart eDirectory.