# Driver Administration Guide

## NetIQ® Identity Manager

**February 2017**

## Legal Notice

For information about NetIQ legal notices, disclaimers, warranties, export and other use restrictions, U.S. Government restricted rights, patent policy, and FIPS compliance, see https://www.netiq.com/company/legal/.

**Copyright (C) 2017 NetIQ Corporation. All rights reserved.**

# Contents


**About this Book and the Library** **7**
**About NetIQ Corporation** **9**

**1 Introduction to Drivers** **11**

Understanding Drivers . . . 11
Activating Drivers . . . 11

**2 Stopping, Starting, or Restarting Drivers** **13**

Stopping, Starting, or Restarting a Driver in Designer . . . 13
Stopping, Starting, or Restarting a Driver in iManager . . . 13

**3 Viewing Version Information** **15**

Viewing a Hierarchical Display of Version Information . . . 15
Viewing and Saving Version Information as a Text File . . . 16
Driver Configuration Files Naming Convention . . . 16

**4 Backing Up Drivers** **17**

Exporting the Driver in Designer . . . 17
Exporting the Driver in iManager . . . 17
Running the Driver in Factory Mode . . . 18

**5 Monitoring Driver Health** **19**

Creating a Driver Health Job . . . 19
Modifying the Driver Health Job's Settings . . . 20
Modifying the Conditions for a Driver Health Configuration . . . 22
Modifying the Actions for a Driver Health Configuration . . . 24
Creating a Custom State . . . 26
Memory Requirements for Driver Health . . . 26

**6 Viewing Driver Statistics** **29**

Viewing Statistics for an Individual Driver . . . 29
Viewing Statistics for a Driver Set . . . 29

**7 Managing Associations between Drivers and Objects** **31**

Associations . . . 31
No-Reference Associations . . . 31
Migration Between Associations . . . 32
- Association Actions in the Main Menu . . . 32
- Association Actions in the Driver Menu . . . 33

Tools for Managing Associations . . . 33
- Inspecting Objects . . . 34
- Inspecting Drivers . . . 35

**8 Managing Driver Cache Files** **37**

Viewing a Transaction . . . 37
Viewing the Out of Band Sync Cache . . . 38
Relocating the Event Cache File . . . 38

**9 Securely Storing Driver Passwords with Named Passwords** **39**

Using Designer to Configure Named Passwords . . . 39
Using iManager to Configure Named Passwords . . . 39
Using Named Passwords in Driver Policies . . . 40
Using the Policy Builder . . . 40
Using XSLT . . . 40
Using the DirXML Command Line Utility to Configure Named Passwords . . . 41
Creating a Named Password in the DirXML Command Line Utility . . . 41
Removing a Named Password in the DirXML Command Line Utility . . . 41

**10 Configuring Java Environment Parameters** **43**

Using iManager to Configure the Java Environment Parameters . . . 43
Using Designer to Configure the Java Environment Parameters . . . 44

**11 Using the DirXML Command Line Utility** **45**

Interactive Mode . . . 45
Command Line Mode . . . 54

**12 Synchronizing Objects** **59**

What Is Synchronization? . . . 59
When Is Synchronization Done? . . . 59
How Does the Identity Manager Engine Decide Which Object to Synchronize? . . . 60
How Does Synchronization Work? . . . 60
Scenario One . . . 61
Scenario Two . . . 62
Scenario Three . . . 63

**13 Association Statistics** **65**

**14 Enabling Out of Band Sync** **67**

Enabling Out of Band Sync Using Designer . . . 67
Enabling Out of Band Sync Using iManager . . . 67
Specifying the Out of Band Sync Status Interval . . . 68

**15 Migrating and Resynchronizing Data** **69**

**16 Configuring Stronger Ciphers for SSL Communication** **71**

Prerequisites . . . 71
Configuring the Settings for Suite B Mode . . . 72
Engine . . . 72
Engine and Remote Loader Communication . . . 72
Engine and Fan-Out Agent Communication . . . 73
Identity Manager Drivers . . . 73

Enabling Stronger Ciphers for SSL Communication . . . 73
Verifying the Suite B Settings . . . 74

**17 Monitoring Identity Manager 75**

Viewing the Monitoring Statistics . . . 76
Monitoring Identity Manager . . . 77
Monitoring Job Statistics . . . 78
Monitoring JVM Statistics . . . 79
Monitoring a Driver Set . . . 81
Monitoring User Application Statistics . . . 81

**18 Rights Needed by a Driver on Identity Vault Objects 85**

**19 Viewing Identity Manager Processes 87**

Adding Trace Levels in Designer . . . 87
Driver . . . 87
Driver Set . . . 88
Adding Trace Levels in iManager . . . 89
Driver . . . 89
Driver Set . . . 89
Capturing Identity Manager Processes to a File . . . 90
Windows . . . 90
UNIX . . . 90
iMonitor . . . 91
Remote Loader . . . 91
Setting Permission for Monitoring Trace Files . . . 92
Working with is-sensitive Attribute . . . 92

**20 Editing Driver Configuration Files 95**

Variables in a Driver Configuration File . . . 95
General Notes . . . 96
Import Driver Notes . . . 98
Flexible Prompting in a Driver Configuration File . . . 98
Viewing the Informal Identity Manager Driver Configuration DTD . . . 100

**21 When and How to Use Global Configuration Values 101**

Using GCVs to Adapt the Driver Configuration File to Changing Environments . . . 101
When Not to Use GCVs . . . 101
When to Use Driver Set GCVs, Driver GCVs, or Global Configuration Objects in Packages . . . 102
Naming Convention for GCVs . . . 102

**22 Synchronizing Permission Changes from the Connected Systems 105**

Planning Overview . . . 106
Prerequisites . . . 106
Understanding the Resource Model . . . 107
Understanding the Components for PCRS . . . 108
Preparing Your Environment . . . 110
Setting Up Administrative User Accounts . . . 110
Setting Up Administrative Passwords . . . 111
Configuring Common Settings GCVs . . . 111
Configuring Custom Entitlements . . . 112

Understanding the PCRS Process . . . 113
Viewing Permission Collection and Reconciliation Service Configuration Objects . . . 116
Troubleshooting Permission Collection and Reconciliation Service Issues . . . 117

**23 Troubleshooting 119**

Identity Manager Treats SYN_TIME values as Signed Integers Instead of Unsigned Integers . . . 119
Using NetIQ Sentinel to Log Identity Manager Events . . . 119
Troubleshooting Driver Processes . . . 119
Driver Shim Errors . . . 119
Identity Manager Driver Errors . . . 122
Java Customization Errors . . . 125
Multiple Sync Events Occur When an Object is Moved in the Master Replica Server . . . 126
Rule Engine Does Not Honor the Mode Specified in the Set or Add Destination Attribute Value . . . 127
Reassociating a Driver Set Object with a Server . . . 128
Association Statistics Tool Displays Incorrect Grouping of Drivers in the Dashboard . . . 128
Association Statistics Tool Displays an Error for No-Reference Associations . . . 128
Unable to Retrieve Application Schema Error . . . 129
Cannot Edit Large Mapping Tables by Using iManager Plug-ins . . . 129

**A Driver Properties 131**

Accessing the Properties . . . 131
Named Passwords . . . 131
Engine Control Values . . . 131
Log Level . . . 135
Driver Image/iManager Icon . . . 135
Security Equals . . . 135
Filter . . . 136
Edit Filter XML . . . 136
Misc/Trace . . . 136
Excluded Objects . . . 137
Driver Health Configuration . . . 137
Driver Manifest . . . 137
Driver Cache Inspector . . . 137
Driver Inspector . . . 137

**B The Cache Flush Parameter 139**

# 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

The library provides the following information resources:

**Identity Manager Setup Guide**

Provides an overview of Identity Manager and its components. This book also provides detailed planning and installation information for Identity Manager.

**Designer Administration Guide**

Provides information about designing, testing, documenting, and deploying Identity Manager solutions in a highly productive environment.

**User Application: Administration Guide**

Describes how to administer the Identity Manager User Application.

**User Application: User Guide**

Describes the user interface of the Identity Manager User Application and how you can use the features it offers, including identity self-service, the Work Dashboard, role and resource management, and compliance management.

**User Application: Design Guide**

Describes how to use the Designer to create User Application components, including how to work with the Provisioning view, the directory abstraction layer editor, the provisioning request definition editor, the provisioning team editor, and the role catalog.

**Identity Reporting Module Guide**

Describes the Identity Reporting Module and how you can use the features it offers, including the Reporting Module user interface and custom report definitions, as well as providing installation instructions.

**Analyzer Administration Guide**

Describes how to administer Analyzer for Identity Manager.

**Identity Manager Driver Guides**

Provide implementation information about specific Identity Manager drivers.

# About NetIQ Corporation

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

## Our Viewpoint

**Adapting to change and managing complexity and risk are nothing new**

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

**Enabling critical business services, better and faster**

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

## Our Philosophy

**Selling intelligent solutions, not just software**

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

**Driving your success is our passion**

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

## Our Solutions

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

# Contacting Sales Support

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

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

# Contacting Technical Support

For specific product issues, contact our Technical Support team.

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

# Contacting Documentation Support

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

# Contacting the Online User Community

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

# 1 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 web page at https://www.netiq.com/documentation/identity-manager-46-drivers/.

## 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.

## Activating Drivers

Identity Manager, Integration Modules (drivers), and the Roles Based Provisioning Module must be activated within 90 days after installation, or they shut down.

# 2 Stopping, Starting, or Restarting Drivers

You might need to start or stop the Identity Manager drivers to ensure that an installation or upgrade process can modify or replace the correct files. It is important to stop a driver before you modify any files for the driver.

## 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 iManager

1. In the **Roles and Tasks** view, click **Identity Manager > Identity Manager Overview**.
2. In the **Search in** field, specify the fully distinguished name of the container where you want to start searching and then click , or click  to browse for and select the container in the tree structure.
3. Click the upper right corner of the driver icon whose status you want to change, then click the appropriate option to stop, start, or restart the driver.

# 3 Viewing Version Information

The Identity Manager engine, the driver shims, and the driver configuration files each contain a separate version number. The Version Discovery Tool in iManager helps you find the versions of the Identity Manager engine and the driver shims versions. The driver configuration files contain their own naming convention.

## Viewing a Hierarchical Display of Version Information

1. In iManager, click **Identity Manager > Identity Manager Overview**, then click **Search** to find the driver sets in the Identity Vault.
2. Click the specific driver set in the list.
3. Click **Driver Set > Version information** on the Driver Set Overview page.
4. View a top-level display of versioning information. The unexpanded hierarchical view displays the following:
   - 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 on each server.
   - Drivers
5. View version information related to servers by expanding the server icon. The expanded view of a top-level server icon displays the following:
   - Last log time
   - Version of Identity Manager that is running on the server
6. View version information related to drivers by expanding the driver icon.

   The expanded view of a top-level driver icon displays the following:
   - The driver name
   - The driver module (for example, com.novell.nds.dirxml.driver.nds.DriverShimImpl)

   The expanded view of a server under a driver icon displays the following:
   - The driver ID
   - The version of the instance of the driver running on that server

# Viewing and Saving Version Information as a Text File

Identity Manager publishes versioning information to a file. You can view this information in text format and save it to a text file on your local or network drive. The textual representation is the same information contained in the hierarchical view.

1. In iManager, click **Identity Manager > Identity Manager Overview**, then click **Search** to find the driver sets in the Identity Vault.
2. Click the specific driver set in the list.
3. Click **Driver Set > Version information** in the Driver Set Overview page.
4. To view the information in the Report Viewer window, click **View**.
5. To save the information to a text file, click **Save As**, specify a file name and location, and click **Save**.

# Driver Configuration Files Naming Convention

The driver configuration file naming convention is:

```
<base name>[-<type>]-IDM<min. engine version>-V<config version>.xml
```

- **Base Name:** The name of the connected system or service the driver provides. For example, Active Directory or Delimited Text.
- **Type:** An additional descriptor for the driver configuration file. If there are multiple configuration files, the type distinguishes among the different files.
- **Minimum Engine Version:** Lists the minimum engine version that the driver can run against. The elements to date are:
  - IDM2_0_0
  - IDM2_0_1
  - IDM2_0_2
  - IDM3_0_0
  - IDM3_0_1
  - IDM3_5_0
  - IDM3_5_1
  - IDM3_6_0
- **Configuration Version:** Specifies the particular driver configuration file version. It is a number that is incriminated with each release of a new driver configuration file.
  - V1
  - V2
  - V11
  - V23

For example:

```
ActiveDirectory-IDM3_6_0-V4.xml
DelimitedText-CSVSample-IDM3_6_0-V2.xml
```

# 4 Backing Up Drivers

After you have created a driver, it is important to create a backup of the driver. You can use Designer or iManager to create an XML file of the driver. The file contains all of the information entered into the driver during configuration. If the driver becomes corrupted, the exported file can be imported to restore the configuration information.

---

**IMPORTANT:** If the driver has been deleted, all of the associations on the objects are 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.

## Exporting the Driver in iManager

1. In iManager, click **Identity Manager > Identity Manager Overview**.
2. Browse to and select the driver set object, then click **Search**.
3. Click the driver icon.
4. Select **Export** in the Identity Manager Driver Overview page.
5. Browse to and select the driver object you want to export, then click **Next**.
6. Select **Export all policies, linked to the configuration or not** or select **Only export policies that are linked to the configuration**, depending upon the information you want to have stored in the XML file.
7. Click **Next**.
8. Click **Save As**, then click **Save**.
9. Browse and select a location to save the XML file, then click **Save**.
10. Click **Finish**.

# 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*.

# 5 Monitoring Driver Health

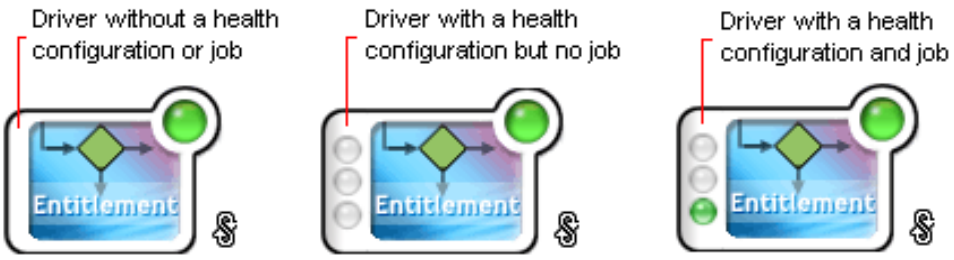
Driver health monitoring allows you to view a driver's current state of health as green, yellow, or red, and to define the actions to perform in response to each of these health states.

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 5-1*** *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 22
- "Modifying the Actions for a Driver Health Configuration" on page 24
- "Creating a Custom State" on page 26

## 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.

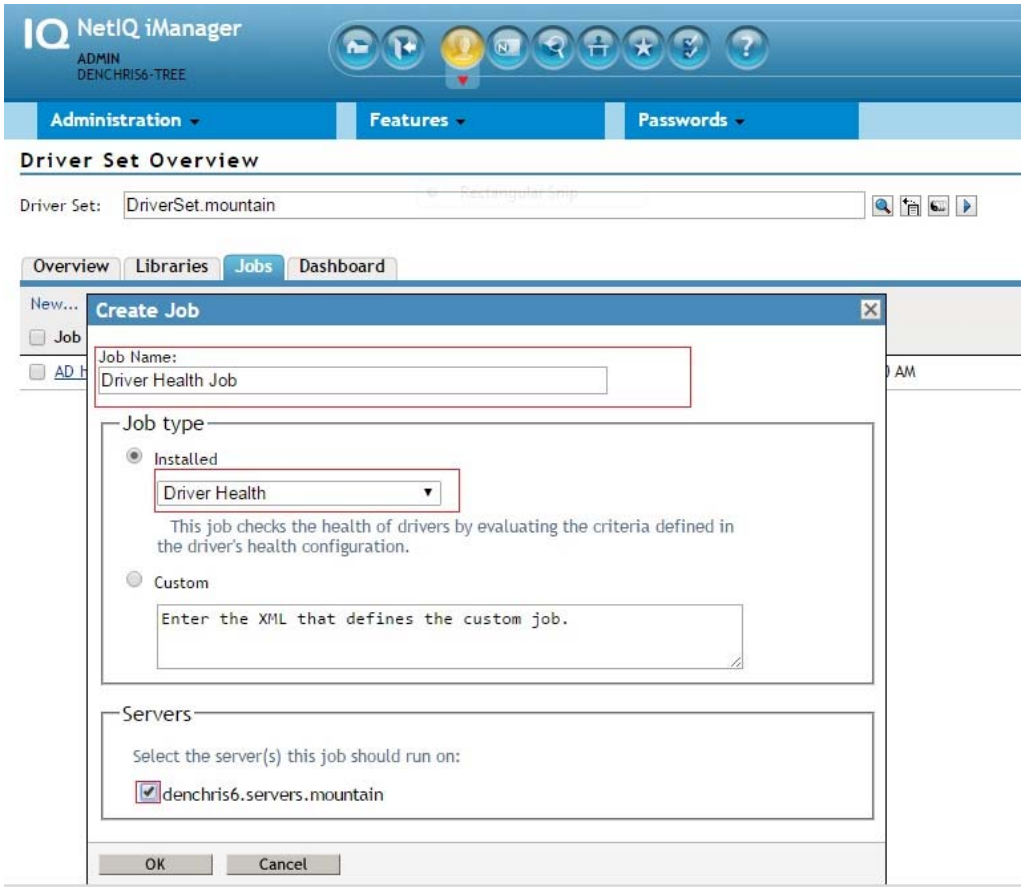
If a Driver Health job does not exist, the Driver Health Configuration page displays a **Run the New Driver wizard and import the Driver Health Job's configuration** prompt. If the page does not display this prompt, the Driver Health job already exists, and you can skip to "Modifying the Conditions for a Driver Health Configuration" on page 22.

**To create a Driver Health job:**

1 In iManager, under Identity Manager Overview, 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 20.

# 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 iManager, under Identity Manager Overview, 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 23conditions, 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 iManager, 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 20 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 24.

# 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 iManager, 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 iManager, 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 22 and "Modifying the Actions for a Driver Health Configuration" on page 24 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.

# 6 Viewing Driver Statistics

You can use NetIQ iManager 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. In iManager, open the Identity Manager Administration page.
2. In the **Administration** list, 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.
3. 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.

   iManager 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 ▸.
4. 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.
5. 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. In iManager, open the Identity Manager Administration page.
2. In the **Administration** list, click **Driver Set Dashboard** to display the Driver Set Query page.
3. 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**.

   iManager 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.

# 7 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 iManager 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. In iManager, 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.

   iManager 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. In iManager, 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.

   iManager 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.

# 8 Managing Driver Cache Files

## Viewing a Transaction

You can use iManager 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. In iManager, 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. In iManager, 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 37.

# 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.

# 9 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 iManager to Configure Named Passwords

1. Locate the driver set or driver where you want to add a named password:
   - **1a** In iManager, open the Identity Manager Administration page.
   - **1b** In the **Administration** list, click **Identity Manager Overview**.
   - **1c** 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.
   - **1d** Click the driver set to open the Driver Set Overview page.
2. To add a named password to a driver set, click the **Driver Set** menu, then click **Edit Driver Set properties**.

   or

   To add a named password to a driver, click the upper right corner of the driver icon, then click **Edit properties**.
3. On the **Identity Manager** tab, click **Named Passwords**.
4. Click **Add**.

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

6 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**.

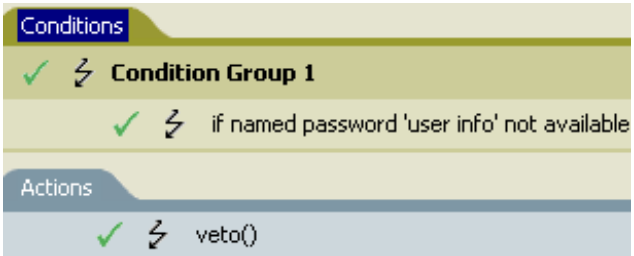
7 Select an action for the **Do** field.

   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 9-1*** *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 11, “Using the DirXML Command Line Utility,” on page 45.
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 11, “Using the DirXML Command Line Utility,” on page 45.
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.

# 10 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 iManager or Designer.

## Using iManager to Configure the Java Environment Parameters

1. In iManager, open the Identity Manager Administration page.
2. Open the properties for the driver set whose parameters you want to configure:
   - **2a** In the **Administration** list, click **Identity Manager Overview**.
   - **2b** If the driver set is not listed on the **Driver Sets** tab, use the **Search In** field to search for and display the driver set.
   - **2c** Click the driver set to open the Driver Set Overview page.
   - **2d** Click the **Driver Set** menu, then click **Edit Driver Set properties**.
3. Click **Misc** 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.

# 11 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: `/usr/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 11-1 on page 45 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 11-1*** *Interactive Mode Options*

| Option | Description |
|---|---|
| **1: Start Driver** | Starts the driver. If there is more than one driver, each driver is listed with a number. Enter the number of the driver to start the driver. |
| **2: Stop Driver** | Stops the driver. If there is more than one driver, each driver is listed with a number. Enter the number of the driver to stop the driver. |

| Option | Description |
|---|---|
| **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 11-2 on page 46. |
| **4: Driver set operations** | Lists the operations available for the driver set. For a list of operations, see Table 11-3 on page 49. |
| **5: Log events operations** | Lists the operations available for logging events through Audit. For a description of these options, see Table 11-6 on page 52. |
| **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 11-2*** *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.<br>◆ 0 - Driver is stopped<br>◆ 1 - Driver is starting<br>◆ 2 - Driver is running<br>◆ 3 - Driver is stopping |
| 4: **Get driver start option** | Lists the current driver start option.<br>◆ 1 - Disabled<br>◆ 2 - Manual<br>◆ 3 - Auto |
| 5: **Set driver start option** | Changes the start option of the driver.<br>◆ 1 - Disabled<br>◆ 2 - Manual<br>◆ 3 - Auto<br>◆ 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).`<br><br>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)`.<br><br>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:`<br><br>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).<br><br>Examples:<br><br>NetWare: `sys:\files\query.xml`<br><br>Windows: `c:\files\query.xml`<br><br>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.<br><br>`Enter filename of XDS command document:`<br><br>Examples:<br><br>Windows: `c:\files\user.xml`<br><br>Linux: `/files/user.xml`<br><br>`Enter name of file for response:`<br><br>Examples:<br><br>Windows: `c:\files\user.log`<br><br>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.<br>`Enter filename of XDS event document:`<br>Examples:<br>Windows: `c:\files\add.xml`<br>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.<br>`Enter filename of XDS event document:`<br>Examples:<br>Windows: `c:\files\add.xml`<br>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).<br>`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 11-4 on page 50 for a description of these options. |
| 14: **Cache operations** | There are five Cache operations. See Table 11-5 on page 51 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 11-3*** *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. |
| **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. |

| Operation | Description |
|---|---|
| **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 11-4 on page 50. |
| **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 11-4*** *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.<br><br>Enter the Remote Loader password, then confirm the password by typing it again. |
| 4: **Clear Remote Loader password** | Clears the Remote Loader password so no Remote Loader password is set on the Driver object. |
| 5: **Set named password** | Allows you to store a password or other pieces of security information on the driver. For more information, see Chapter 9, “Securely Storing Driver Passwords with Named Passwords,” on page 39.<br><br>There are four prompts to fill in:<br><br>◆ **Enter password name:**<br>◆ **Enter password description:**<br>◆ **Enter password:**<br>◆ **Confirm password** |

| Operation | Description |
|---|---|
| 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).`<br><br>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:<br><br>◆ Driver Object password:<br>◆ Application password:<br>◆ Remote loader password:<br><br>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 11-5*** *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. |
| 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.<br><br>◆ **Enter position token** (default=0):<br>◆ **Enter maximum transactions records to return** (default=1):<br>◆ **Enter name of file for response:** |
| 4: **Delete cached transactions** | Deletes the transactions stored in the cache.<br><br>◆ **Enter position token** (default=0):<br>◆ **Enter event-id value of first transaction record** to delete (optional):<br>◆ **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 have 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 11-6*** *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 11-7 on page 52 for a list of these options.<br><br>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 11-7 on page 52 for a list of these options.<br><br>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 11-7*** *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 |

| Options |
| --- |
| 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 |
| 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 |

| Options |
| --- |
| 49: Clear SSO Credential |
| 50: Set SSO Passphrase |
| 51: User defined IDs |
| 99: Accept checked items |

***Table 11-8*** *Job Operations*

| Options | Description |
| --- | --- |
| 1: **Get available job definitions** | Allows you to select an existing job.<br>`Enter the driverset number or the driver number:`<br>`Do you want to filter the job definitions by containment?` Enter Yes or No<br>`Enter name of the file for response:`<br>Examples:<br>Windows: `c:\files\user.log`<br>Linux: `/files/user.log` |
| 2: **Operations on specific job object** | Allows you to perform operations for a specific job.<br>`Enter the job number:`<br>The following list of options appears:<br>**1: Send job update notification**<br>**2: Start job**<br>**3: Abort running job**<br>**4: Get job state**<br>**5: Check job configuration**<br>**6: Passwords operations**<br>**99: Exit** |

# Command Line Mode

The command line mode allows you to use script or batch files. Table 11-9 on page 55 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 11-9*** *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. |
| -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.<br><br>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.<br><br>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.<br><br>Specify the XDS command document as the input file.<br><br>Examples:<br><br>NetWare: `sys:\files\user.xml`<br><br>Windows: `c:\files\user.xml`<br><br>Linux: `/files/user.log`<br><br>Specify the output filename to see the results.<br><br>Examples:<br><br>NetWare: `sys:\files\user.log`<br><br>Windows: `c:\files\user.log`<br><br>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. |
| -setlogevents *<dn>* *<integer ...>* | Sets Audit log events on the driver. The integer is the option of the item to log. See Table 11-7 on page 52 for the list of the integers to enter. |

| Option | Description |
|---|---|
| -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 11-10 on page 58 contains other values for specific command line options.

*Table 11-10 Command Line Option Values*

| Command Line Option | Values |
|---|---|
| -getstate | 0- stopped<br>1- starting<br>2- running<br>3- shutting down<br>11- get schema<br>Anything else that is returned is an error.<br>The getstate option returns the state of the driver. It does not show the state.<br>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<br>1- manual<br>2- auto<br>Anything else that is returned is an error. |
| -getcachelimit | 0- unlimited<br>Anything else that is returned is an error. |
| -getjobrunningstate | 0- stopped<br>1- running<br>Anything else that is returned is an error. |
| -getjobenabledstate | 0- disabled<br>1- enabled<br>2- configuration error<br>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). |

# 12 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 iManager Identity Manager Driver 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 60.

# 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 12-1 on page 62, Table 12-2 on page 63, and Table 12-3 on page 64.

   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.

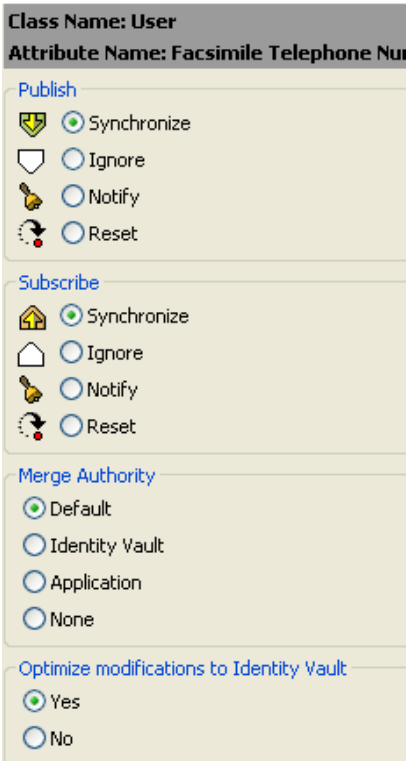
There are three different combinations of selected items in the filter, and each one creates a different output.


- "Scenario One" on page 61
- "Scenario Two" on page 62
- "Scenario Three" on page 63


## Scenario One

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

***Figure 12-1*** *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.

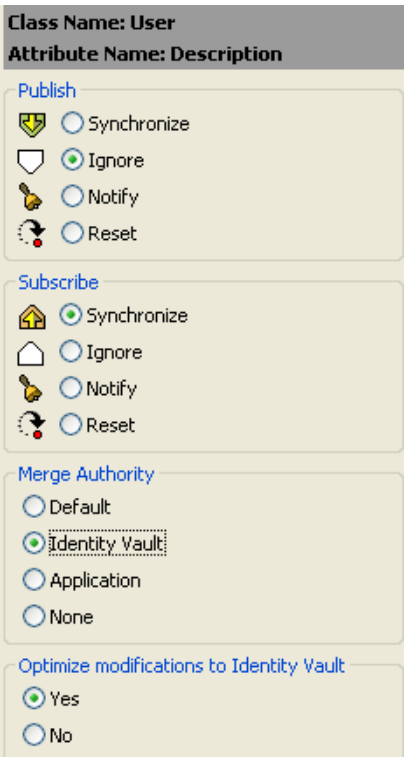
***Table 12-1*** *Output of Scenario One*

| | **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<br>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 12-2*** *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.

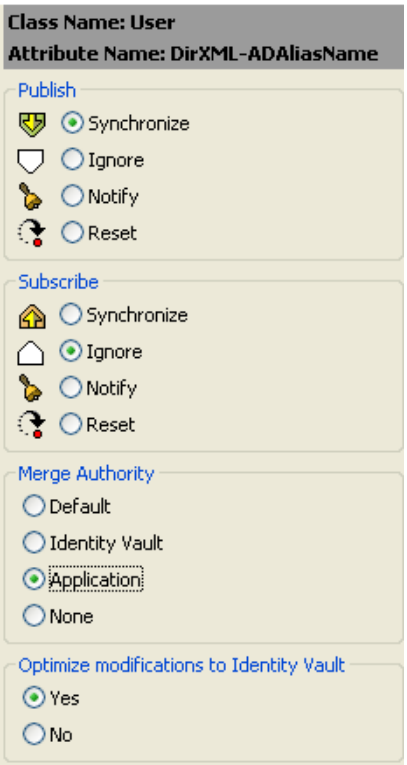
***Table 12-2*** *Output of Scenario Two*

| | **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 12-3*** *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 12-3*** *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 |

# 13 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 iManager. 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 iManager to Configure the Java Environment Parameters" on page 43.

---

**To view the association statistics:**

1. Log in to iManager.
2. Click to view the Identity Manager administration page.

   ---

   **NOTE:** Make sure that you enable pop-ups on your browser before you run the Association Statistics tool.

   ---

3. In the **Administration** section, click **Association Statistics**.

   Alternatively, you can access **Association Statistics** from the **Roles and Tasks** page. Expand **Identity Manager** node and click **Association Statistics**.
4. In the **Driverset** field, click to browse and select the driverset on which you want to run the association statistics.
5. Click **OK**. The association count displays the previously computed result.

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

   iManager considers groups and organization units as system managed objects. iManager 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.
6. Click **Recompute** and then click **OK** to obtain the updated results.

   When a driver is disabled on the server, iManager does not display the driver in the dashboard.
7. To view the association count for drivers associated with a different server, select the server from the **Driver Information From** drop-down list.

8 Click **Export Association Statistics** to export the system details and association count details for the drivers associated with the server.

9 To export the objects associated with a specific driver, click ⬇ 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, iManager displays all the association counts in the dashboard. However, if you choose to export the objects in a file, iManager exports only the unique objects.

---

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

# 14 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.

## Enabling Out of Band Sync Using iManager

Perform the following steps:

1. In iManager, open the Identity Manager Administration page.
2. In the **Administration** list, click **Identity Manager Overview** to display the Identity Manager Overview page.
3. In the **Driver Sets** list, select the desired driver set to display the Driver Set Overview page.
4. Select the desired driver to display the Driver Overview page.
5. Click the **Driver Filter** icon to open the Filter page.
6. Click the attribute for which you want to enable Out of Band Sync, and then set **Perform Out of Band Sync (Subscriber)** to **Yes**. By default, it is set to **No**.

# 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 38.

# 15 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 iManager:

- **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 iManager, 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 12, “Synchronizing Objects,” on page 59.

# 16 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. |
| 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. |

| Requirement | Description |
| --- | --- |
| Cipher specification | The following ciphers are supported for Suite B mode:<br>◆ TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256<br>◆ TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384<br>To use ciphers with stronger signature and hash algorithms, the certificates of server key file must contain similar or stronger signature and hash algorithms.<br>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.<br>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 iManager.

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 Understanding the Configuration Parameters for the Remote Loader in the *NetIQ Identity Manager Setup Guide*.

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 Fan-Out Implementation Guide*.

## 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 iManager and 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 in the *NetIQ Driver for eDirectory Implementation Guide*.

**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>
```

# 17 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.

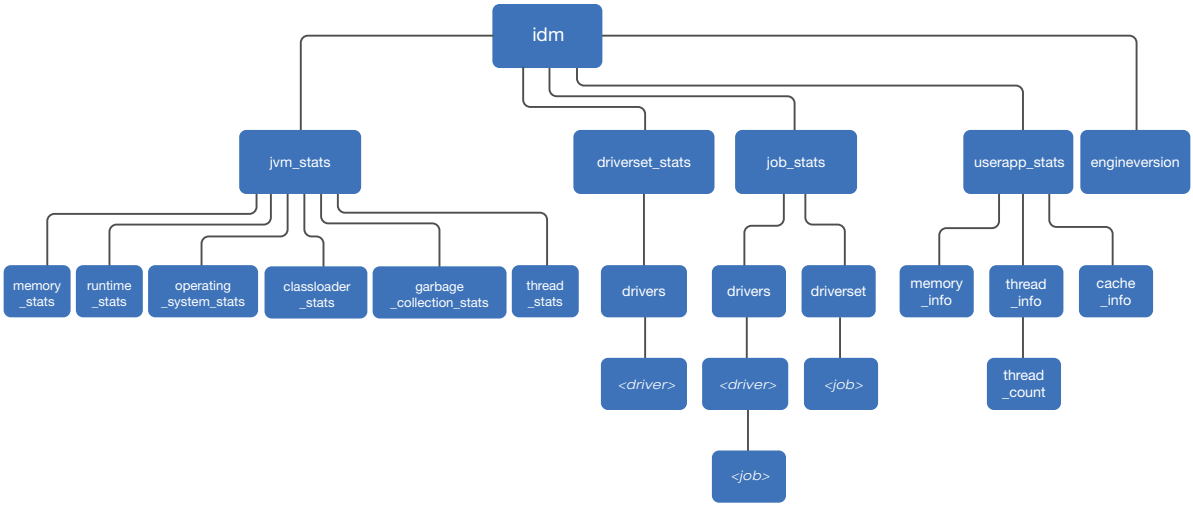
You can obtain the monitoring data by issuing a search request with a search base of `cn=idm,cn=monitor` on the Identity Vault. Using this method provides several advantages. The search is quick and can be embedded in a script for gathering monitoring data at regular intervals. Also, you can consolidate 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 76.

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.<br>◆ memory_stats: Provides information about the usage of heap and non-heap memory.<br>◆ thread_stats: Provides information for all live threads, such as name, state, and the count of active threads.<br>◆ runtime_stats: Provides information about the JVM environment, such as start time, uptime, and system properties.<br>◆ operating_system_stats: Provides information about the underlying operating system on which Identity Manager is installed, such as name, version, processor, and architecture.<br>◆ classloader_stats: Provides information about the class loader of the JVM, such as number of loaded classes, unloaded classes, and total number of classes.<br>◆ 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:<br>◆ driverset: Provides information about a specific job or all jobs configured for a driver set.<br>◆ 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 for the User Application:<br>◆ MemoryInfo: Provides memory related information such as system memory and memory consumed by the JVM.<br>◆ ThreadInfo: Provides information about the CPU-intensive threads and returns the list of top threads that cause heavy utilization of the CPU.<br>  ◆ ThreadCount: Provides the current number of live threads.<br>◆ CacheInfo: 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.<br>To obtain the User Application information, ensure that the following prerequisites are met:<br>◆ Common Setting Advanced Edition package is installed on driver set containing the User Application driver<br>  This package contains User Application Provisioning Services URL (`UAProvURL`) and User Application Provisioning Services Administrator (`UAProvAdmin`) GCVs.<br>◆ `UAProvURL` and `UAProvAdmin` GCVs have the correct values.<br>For more information, see the *NetIQ Identity Manager - Administrator's Guide to the Identity Applications*. |

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 78
- "Monitoring JVM Statistics" on page 79
- "Monitoring a Driver Set" on page 81
- "Monitoring User Application Statistics" on page 81

## 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=monitor
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

You can monitor information about the health of the User Application such as currently running threads, memory consumption, and cache information.

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 `javamemorypool`. 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 on the following entry:

`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`

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`

A sample output looks like this:

```
dn:
cn=RUNNABLE,cn=threadStateCounts,cn=thread_count,cn=thread_info,cn=userapp_stats,c
n=idm,cn=Monitor
count: 21
objectclass: Top
objectclass: extensibleObject

dn:
cn=TIMED_WAITING,cn=threadStateCounts,cn=thread_count,cn=thread_info,cn=userapp_st
ats,cn=IDM,cn=Monitor
count: 46
objectclass: Top
objectclass: extensibleObject

dn:
cn=WAITING,cn=threadStateCounts,cn=thread_count,cn=thread_info,cn=userapp_stats,cn
=IDM,cn=Monitor
count: 51
objectclass: Top
objectclass: extensibleObject

dn: cn=SSPR--1-LocalDBLogger-writer-pool-161-thread-
1,cn=topMostCPUConsumingThread,cn=thread_count,cn=thread_info,cn=userapp_stats,cn=
IDM,cn=Monitor
state: TIMED_WAITING
objectclass: Top
objectclass: extensibleObject
```

A sample LDAP search output is below.

```
dn: cn=cache_info,cn=userapp_stats,cn=idm,cn=monitor
cachearraysize: 19
objectclass: Top
objectclass: extensibleObject

dn:
cn=Authorization.AdminLevelsCacheHolder,cn=cacheHolders,cn=cache_info,cn=userapp_s
tats,cn=idm,cn=monitor
objectcount: 1
objectclass: Top
objectclass: extensibleObject

dn:
cn=DirectoryAbstractLayerDefinitions,cn=cacheHolders,cn=cache_info,cn=userapp_stat
s,cn=idm,cn=monitor
objectcount: 105
objectclass: Top
objectclass: extensibleObject

dn:
cn=Workflow.Model.Request,cn=cacheHolders,cn=cache_info,cn=userapp_stats,cn=idm,cn
=monitor
objectcount: 0
objectclass: Top
objectclass: extensibleObject

dn: cn=MEMORY_INFO,cn=userapp_stats,cn=idm,cn=monitor
commitedvirtualmemory: 2972792
objectclass: Top
objectclass: extensibleObject

dn: cn=Code
Cache,cn=javaMemoryPool,cn=memory_info,cn=userapp_stats,cn=idm,cn=monitor
freeMemory: 163010
usedMemory: 82750
maxMemory: 245760
objectclass: Top
objectclass: extensibleObject

dn:
cn=HeapMemoryUsage,cn=jvmMemoryUsage,cn=memory_info,cn=userapp_stats,cn=idm,cn=mon
itor
freememory: 514910
usedmemory: 498722
maxmemory: 1013632
objectclass: Top
objectclass: extensibleObject

dn:
cn=NonHeapMemoryUsage,cn=jvmMemoryUsage,cn=memory_info,cn=userapp_stats,cn=idm,cn=
monitor
freeMemory: 0
usedMemory: 280211
maxMemory: 0
objectclass: Top
objectclass: extensibleObject

dn:
cn=physicalMemory,cn=systemMemoryUsage,cn=memory_info,cn=userapp_stats,cn=idm,cn=m
onitor
```

```
freeMemory: 888852
usedMemory: 2972036
maxMemory: 3860888
objectclass: Top
objectclass: extensibleObject

dn:
cn=swapMemory,cn=systemMemoryUsage,cn=memory_info,cn=userapp_stats,cn=idm,cn=monit
or
freeMemory: 688020
usedMemory: 1415272
maxMemory: 2103292
objectclass: Top
objectclass: extensibleObject

dn:
cn=RUNNABLE,cn=threadStateCounts,cn=thread_info,cn=userapp_stats,cn=idm,cn=monitor
count: 27
objectclass: Top
objectclass: extensibleObject

dn:
cn=ContainerBackgroundProcessor[StandardEngine[Catalina]],cn=topMostCPUConsumingTh
read,cn=thread_info,cn=userapp_stats,cn=idm,cn=monitor
state: TIMED_WAITING
objectclass: Top
objectclass: extensibleObject

dn: cn=SSPR--1-LocalDBLogger-writer-pool-161-thread-
1,cn=topMostCPUConsumingThread,cn=thread_info,cn=userapp_stats,cn=idm,cn=monitor
state: TIMED_WAITING
objectclass: Top
objectclass: extensibleObject
```

# 18 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)

# 19 Viewing Identity Manager Processes

To view Identity Manager processing events, use DSTrace. You use this only during testing and troubleshooting Identity Manager. Running DSTrace 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 DSTrace, you add values to the driver set and the drivers. You can do this in Designer and iManager.

## Adding Trace Levels in Designer

You can add trace levels to each driver or to the driver set.

### Driver

1. In an open project in Designer, select the driver in the Outline view.
2. Right-click and select **Properties**, then click **8. Trace**.
3. Set the parameters for tracing, then click **OK**. See Table 19-1 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 DSTrace log.

***Table 19-1*** *Driver Trace Parameters*

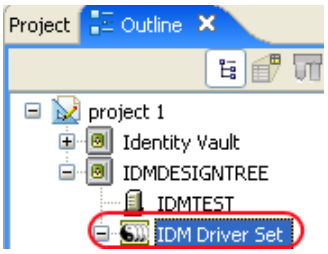
| Parameter | Description |
|---|---|
| Trace level | As the driver trace level increases, the amount of information displayed in DSTrace increases.<br><br>Trace level one shows errors, but not the cause of the errors. If you want to see password synchronization information, set the trace level to five.<br><br>If you select **Use setting from Driver Set**, the value is taken from the driver set. |
| Trace file | Specify a file name and location of where the Identity Manager information is written for the selected driver.<br><br>If you select **Use setting from Driver Set**, the value is taken from the driver set. |
| Trace file encoding | The trace file uses the system's default encoding. You can specify another encoding if desired. |

| Parameter | Description |
|---|---|
| Trace file size limit | Allows you to set a limit for the Java trace file.<br><br>**WARNING:** If you set the file size to unlimited, the file continues to grow in size, and consumes all the disk space. Set the file size to unlimited, only for a limited time and carefully monitor the free disk space.<br><br>**NOTE:** The trace file is created in multiple files. Identity Manager automatically divides the maximum file size by ten and creates ten separate files. The combined size of these files equals the maximum trace file size.<br><br>If you select **Use setting from Driver Set**, the value is taken from the driver set. |
| 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. |

# Driver Set

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 In an open project in Designer, select the driver set in the **Outline** view.




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

3 Set the parameters for tracing, then click **OK**. See Table 19-2 on page 88 for more information about the driver set trace parameters.

If you set the trace level on the driver set, all drivers appear in the DSTrace logs.

***Table 19-2*** *Driver Set Trace Parameters*

| Parameter | Description |
|---|---|
| Driver trace level | As the driver trace level increases, the amount of information displayed in DSTrace increases.<br><br>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. |
| XSL trace level | DSTrace displays XSL events. Set this trace level only when troubleshooting XSL style sheets. If you do not want to see XSL information, set the level to zero. |
| Java debug port | Allows developers to attach a Java debugger. |

| Parameter | Description |
| --- | --- |
| Java trace file | 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.<br><br>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.<br><br>**WARNING:** If you set the file size to unlimited, the file continues to grow in size, and consumes all the disk space. Set the file size to unlimited, only for a limited time and carefully monitor the free disk space<br><br>**NOTE:** The trace file is created in multiple files. Identity Manager automatically divides the maximum file size by ten and creates ten separate files. The combined size of these files equals the maximum trace file size. |

# Adding Trace Levels in iManager

You can add trace levels to each driver or to the driver set.

## Driver

1. In iManager, open the Identity Manager Administration page.
2. Open the properties for the driver set that contains the driver you want to configure:
   - **2a** In the **Administration** list, click **Identity Manager Overview**.
   - **2b** If the driver set is not listed on the **Driver Sets** tab, use the **Search In** field to search for and display the driver set.
   - **2c** Click the driver set to open the Driver Set Overview page.
3. Click the upper right corner of the driver, then click **Edit properties**.
4. Select the **Misc** tab for the driver.
5. Set the parameters for tracing, then click **OK**. See Table 19-1 on page 87 for information.

## Driver Set

1. In iManager, open the Identity Manager Administration page.
2. Open the properties for the driver set whose parameters you want to configure:
   - **2a** In the **Administration** list, click **Identity Manager Overview**.
   - **2b** If the driver set is not listed on the **Driver Sets** tab, use the **Search In** field to search for and display the driver set.
   - **2c** Click the driver set to open the Driver Set Overview page.
   - **2d** Click the **Driver Set** menu, then click **Edit Driver Set properties**.

3 Select the **Misc** tab for the driver set.

4 Set the parameters for tracing, then click **OK**. See Table 19-2 on page 88 for information about these parameters.

# Capturing Identity Manager Processes to a File

Identity Manager processes are saved to a file by using a parameter on the driver or through DSTrace. The parameter on the driver is the Trace file parameter.

The driver processes that are captured through DSTrace are the processes that occur on the Identity Manager engine. If you use the Remote Loader, you need to capture a trace on the Remote Loader at the same time as you are capturing the trace on the Identity Manager engine.

The following methods help you capture and save Identity Manager processes through DSTrace on different OS platforms.

## Windows

1 Open the Control Panel > **NDS Services** > **dstrace.dlm**, then click **Start**.

2 In the NDS Server Trace Utility window, select **Edit** > **Options**, then click **Clear All**.

   This clears all of the default flags.

3 Select **DirXML** and **DirXML Drivers**.

4 Click **OK**.

5 Select **File > New**.

6 Specify the filename and location of where you want the DSTrace information saved, then click **Open**.

7 Wait for the event to occur.

8 Select **File > Close**.

   This stops the information from being written to the log file.

9 Open the file in a text editor and search for the event or the object you modified.

## UNIX

1 Enter `ndstrace` to start the ndstrace utility.

2 Enter `set ndstrace=nodebug`

   This turns off all trace flags currently set.

3 Enter `set ndstrace on`

   This displays trace messages to the console.

4 Enter `set ndstrace file on`

   This captures trace messages to the file `ndstrace.log` in the directory where eDirectory is installed. By default it is `/var/nds`.

5 Enter `set ndstrace=+dxml`

   This displays the Identity Manager events.

6 Enter `set ndstrace=+dvrs`

   This displays the Identity Manager driver events.

7 Wait for the event to occur.

8 Enter `set ndstrace file off`

   This stops the logging of information to the file.

9 Enter `exit` to quit the ndstrace utility.

10 Open the file in a text editor. Search for the event or the object that was modified.

## iMonitor

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

- `NDSIMON.DLM` Runs on Windows
- `ndsimonitor` Runs on UNIX

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

   Port 8008 is the default port.

2 Enter a username and password with administrative rights, then click **Login**.

3 Select **Trace Configuration** on the left side.

4 Click **Clear All**.

5 Select **DirXML and DirXML Drivers**.

6 Click **Trace On**.

7 Select **Trace History** on the left side.

8 Click the document with the **Modification Time** of **Current** to see a live trace.

9 Change the **Refresh Interval** if you want to see information more often.

10 Select **Trace Configuration** on the left side, then click **Trace Off** to turn the tracing off.

11 You can view the trace history by selecting Trace History. The files are distinguished by their time stamps.

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

- Windows: `Drive_letter:\Novell\NDS\ndsimon\dstrace\*.htm`
- UNIX/Linux: `/var/nds/dstrace/*.htm`

## Remote Loader

You can capture the events that occur on the computer running the Remote Loader service.

1 Launch the Remote Loader Console by clicking the icon.

2 Select the driver instance, then click **Edit**.

3 Set the **Trace Level** to 3 or above.

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.

You can also enable tracing from the command line by using the switches in the following table. For more information, see "Configuring the Remote Loader and Drivers" in the *NetIQ Identity Manager Setup Guide*.

*Table 19-3 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.<br><br>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.<br><br>Example:<br><br>`-tracefile c:\temp\trace.txt`<br><br>or<br><br>`-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.<br><br>The size parameter is the number of bytes. Specify the size by using the suffixes K, M, or G for kilobytes, megabytes, or gigabytes.<br><br>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.<br><br>Example: `-tracefilemax 1000M or -tfm 1000M` |

# 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>
```

# 20 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 "Managing 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

For the iManager plug-ins, 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 98.

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 98.
- **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:

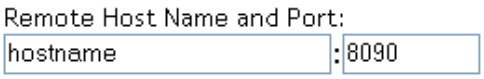
- "General Notes" on page 96
- "Import Driver Notes" on page 98

## General Notes

Review the following general notes:

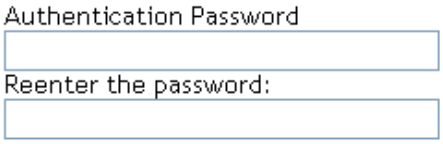
- A `variable-decl` can contain `text-var` but not `node-var`. It can contain `variable-refs` as long as the order they are resolved is taken into account.
- If a `variable-decl` contains an optional prompt attribute and an optional prompt-type attribute and does not contain an optional `browse="yes"` attribute setting, the prompt-type is treated as follows:
  - A prompt-type of `"ipa"` results in two edit fields. See Figure 20-1 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 20-1*** *Two Edit Fields*

  - A prompt-type of `"password"` results in two password edit fields. See Figure 20-2 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 20-2*** *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 98 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.

# 21 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
```

# 22 Synchronizing Permission Changes from the Connected Systems

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.

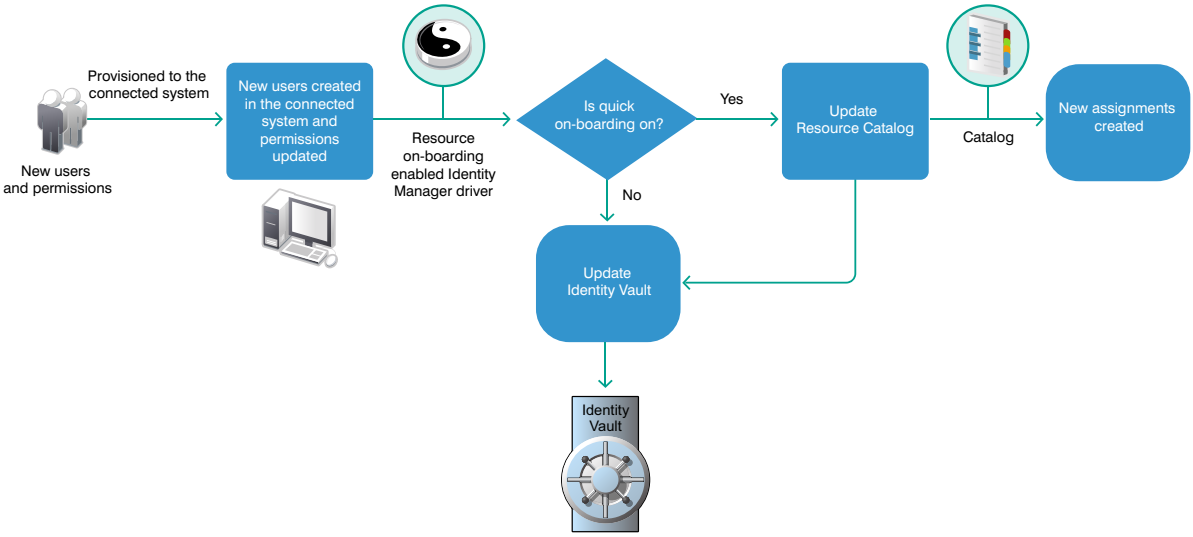
For a newly installed driver with this package content, you can migrate users and groups (for example, Active Directory) into the Identity Vault, which updates the Resource Catalog with the current state of the connected system. For example, if `User1` and `User2` are part of `Group1` with the required 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 106
- "Preparing Your Environment" on page 110
- "Viewing Permission Collection and Reconciliation Service Configuration Objects" on page 116
- "Troubleshooting Permission Collection and Reconciliation Service Issues" on page 117


# 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

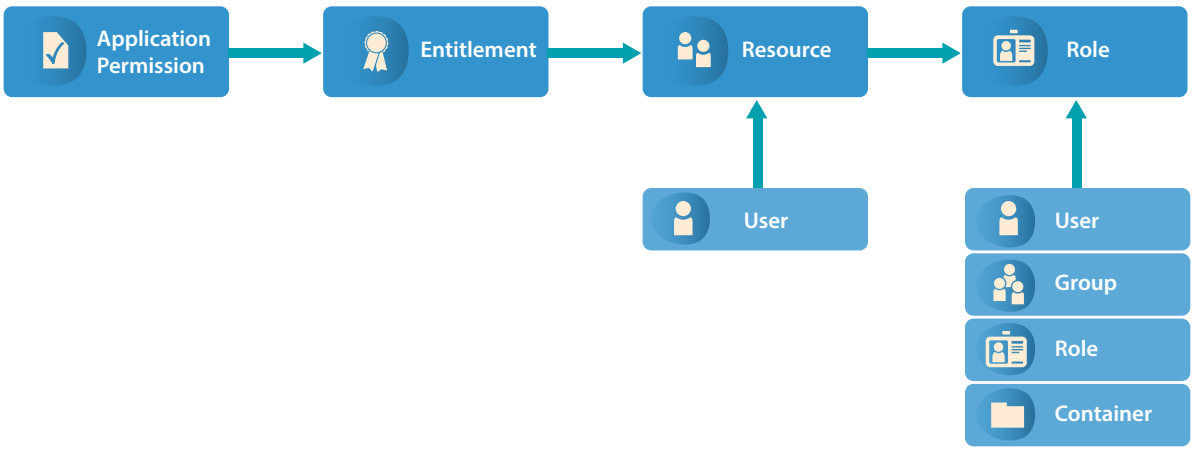
In Identity Manager, a resource is any digital entity such as a user account, computer, or database that a business user needs to be able to access.

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 108
- "Identity Manager Policy Objects" on page 108
- "PermissionOnboarding Job Object" on page 109
- "Mapping Table Objects" on page 109
- "Understanding the CSV File Format" on page 109


## 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 111. For information about `PermissionOnboarding` job, see "PermissionOnboarding Job Object" on page 109.

## 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 "Password Management Configuration" in 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 113.

# 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 109.
- **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**.

  **NOTE:** In case you changed the value of this parameter after the driver starts, delete the associated entitlement object and the resource, and then restart the driver.

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.

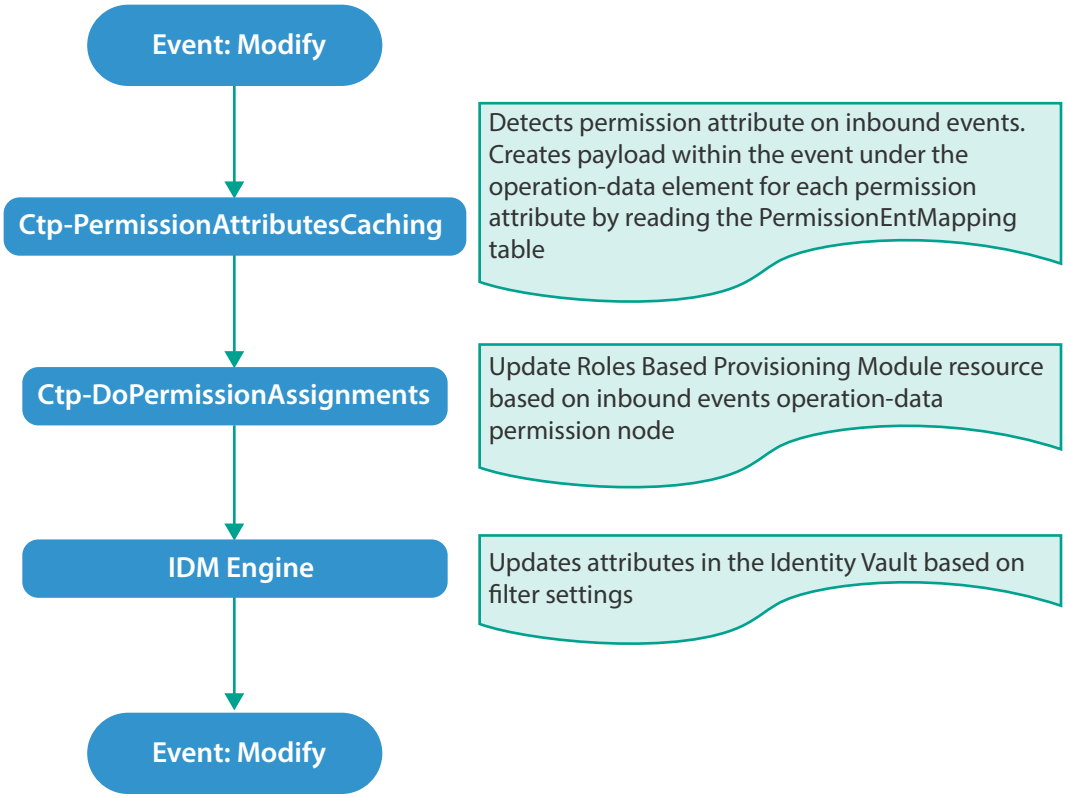
   Now the driver is ready to synchronize the connected system permission changes to the User Application.

For more information, see "Viewing Permission Collection and Reconciliation Service Configuration Objects" on page 116.

## How Permissions Are Reconciled

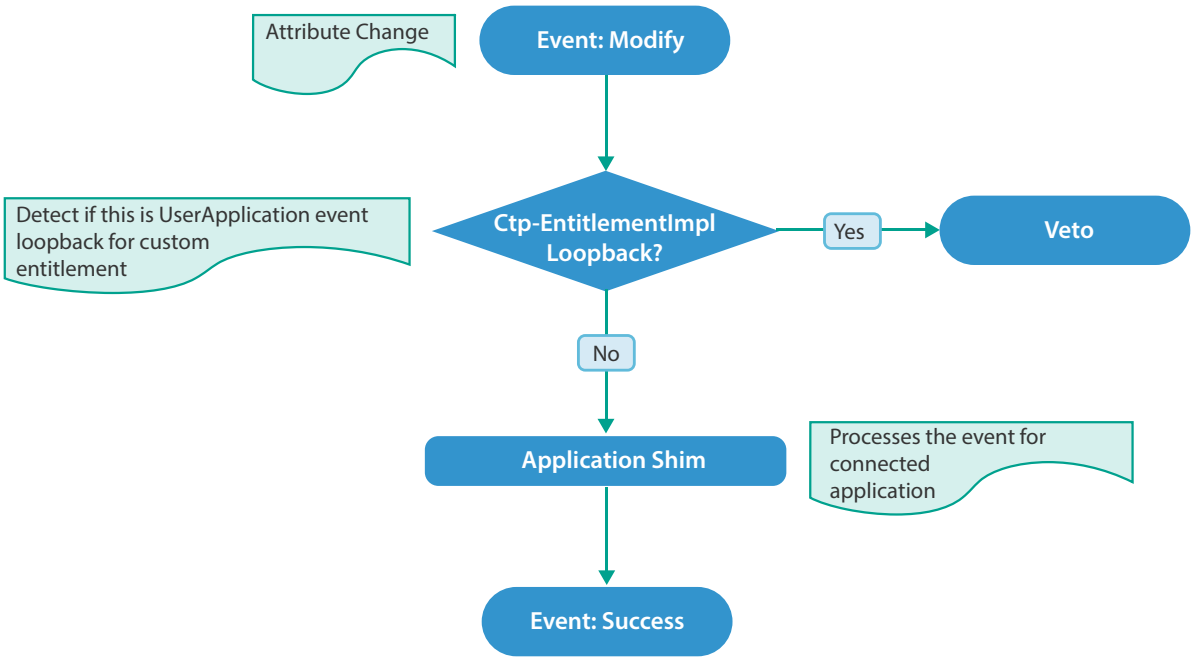
The below examples will help you understand how permissions are reconciled in different scenarios.

**Use Case 1:** When the Publisher channel has an event with permission changes.



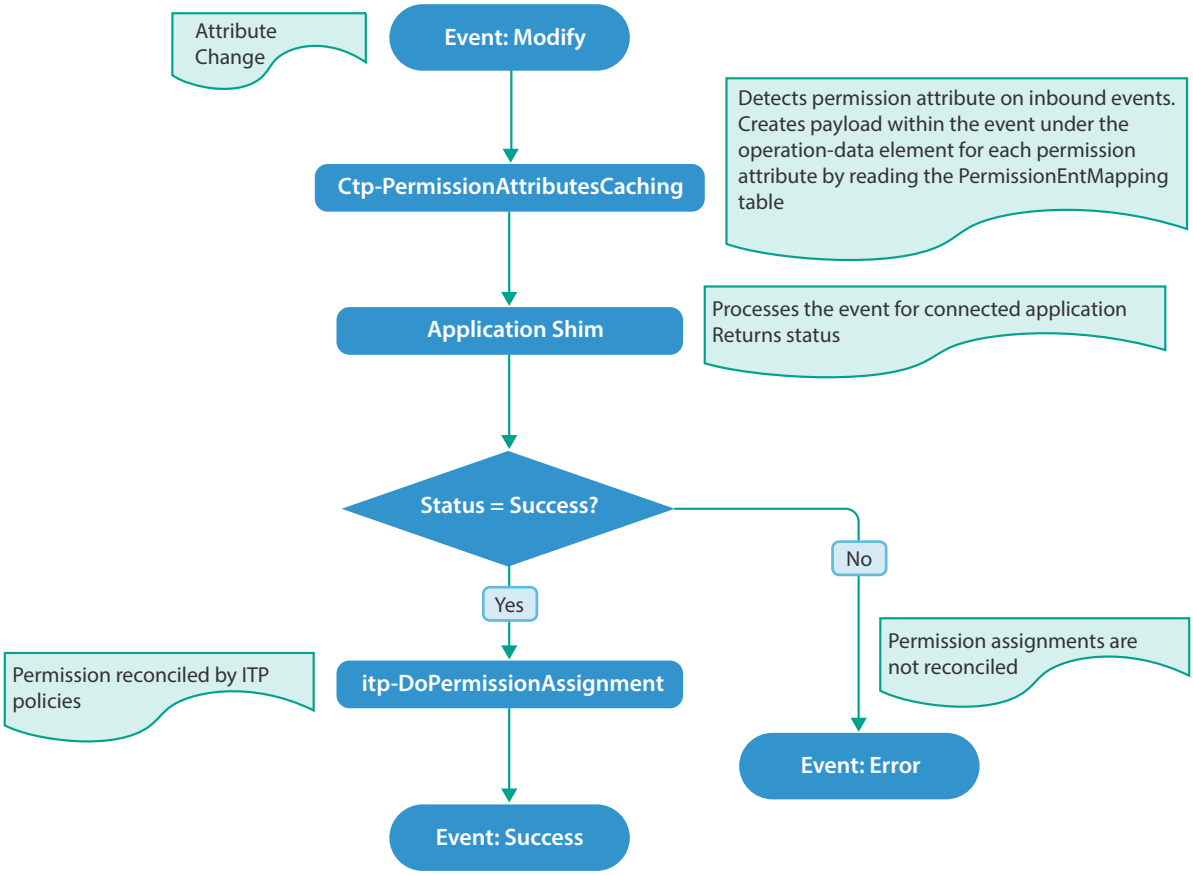

**Use Case 2:** When a permission attribute changes in the Identity Vault.

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. In iManager, 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. In iManager, 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.

# 23 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, 4.7.1 and 4.7 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.

## 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 Reporting Guide for Sentinel*.

## Driver Connects to The Default Port Instead of The Custom Ports

**Issue:** The driver ignores the custom port number specified in the subscriber and publisher parameters and continues to connect to the default 8192 port.

**Workaround:** To support the custom ports for local and remote tree, ensure that the following format is followed in the **Authentication context** field in iManager:

```
localTreeIpAdress:port:RemoteTreeIpAdress:port
```

## 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 19, “Viewing Identity Manager Processes,” on page 87.

# 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 either iManager or 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/Solaris/AIX:**

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.

   In eDirectory 8.8.x, the files are present 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 124error.

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 123.

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:

  http://www.novell.com/communities/node/2332/supportconfig-linux (http://www.novell.com/communities/node/2332/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. In iManager, open the Identity Manager Administration page.
2. In the **Administration** list, click **Identity Manager Overview**.
3. In the **Search in** field, specify the fully distinguished name of the container where you want to start searching and then click ▶, or click the browse icon to browse for and select the container in the tree structure.
4. Click the driver set object that you want to reassociate with a server.
5. On the **Overview** tab, click **Servers**.
6. Click **Remove server**.
7. Click **Add server**.

---

**NOTE:** When you reassociate a driver set object with a server, all drivers are disabled, and all passwords are cleared.

---

# Association Statistics Tool Displays Incorrect Grouping of Drivers in the Dashboard

If a driver is not initialized, iManager displays the driver in the *IDM Driver (Other)* group on the association statistics dashboard. This is because the driver configuration does not populate the group ID for the driver.

To work around this issue, make sure that the driver is initialized with the connected system.

# Association Statistics Tool Displays an Error for No-Reference Associations

If you attempt to calculate no-reference associations using the association statistics tool, iManager displays an `ERR_FAILED_AUTHENTICATION` error. This issue is randomly observed.

To work around this issue, run the Association Statistics job again.

# Unable to Retrieve Application Schema Error

**Issue:** Some Identity Manager drivers are not intended to connect to an application. For such drivers, the engine does not retrieve an application schema because there is no connected application. Therefore, the driver shim returns an empty schema to the engine. The engine considers it as a missing schema and displays the following warning:

```
DirXML Log Event -------------------
     Driver:   \Org_Tree1\system\services\idm\Driverset1\Timed Job
     Status:   Warning
     Message: Code(-8001) Unable to retrieve application schema.
[11/10/17 15:50:58.673]:Timed Job ST:Reading driver information from the \
Org_Tree1\system\services\idm\Driverset1\Timed Job object.
[11/10/17 15:50:58.675]:Timed Job ST:Loading Java shim
com.novell.nds.dirxml.driver.nulldriver.NullDriverShim.
```

**Workaround:** Ignore this warning for the drivers that are not connected to an external application.

# Cannot Edit Large Mapping Tables by Using iManager Plug-ins

**Issue:** When you modify a large mapping table that has approximately 8000 entries by using iManager plug-ins for Identity Manager, iManager reports a Java exception in the `catalina.out` file and logs you out of the application.

This issue is not reported in Designer.

**Workaround:** Remove the post request size and parameter limit from Tomcat's `server.xml` file.

1. Navigate to the `server.xml` file. For example, `/var/opt/novell/tomcat8/conf` or `C:\Program Files\Novell\Tomcat\conf\`.
2. Add the following attributes under the `<Connector>` entry in the file.

   ```
   maxParameterCount="-1"
   maxPostSize="-1"
   ```

3. Restart the iManager service.

# A 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 iManager. If a field is different in Designer for Identity Manager, it is marked with a Designer icon.

## Accessing the Properties

**In iManager:**

1. In iManager, open the Identity Manager Administration page.
2. Open the driver set that contains the driver whose properties you want to edit:
   - **2a** In the **Administration** list, click **Identity Manager Overview**.
   - **2b** If the driver set is not listed on the **Driver Sets** tab, use the **Search In** field to search for and display the driver set.
   - **2c** Click the driver set to open the Driver Set Overview page.
3. Locate the driver icon, then click the upper right corner of the driver icon to display the **Actions** menu.
4. Click **Edit Properties** to display the driver's properties page.

**In Designer:**

1. Open a project in the Modeler.
2. Right-click the driver icon or line, then select click **Properties** > **Driver Configuration**.

## 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.

For more information about named passwords, see Chapter 9, "Securely Storing Driver Passwords with Named Passwords," on page 39.

## 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. |
| **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.<br><br>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.<br><br>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.<br><br>**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.<br><br>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. |

| Option | Description |
| --- | --- |
| **Write pending associations** | This control determines whether the Identity Manager engine writes a pending association on an object during Subscriber channel processing.<br><br>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. |
| **Use password event values** | This control determines the source of the value reported for the nspmDistributionPassword attribute for Subscriber channel Add and Modify events.<br><br>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.<br><br>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.<br><br>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.<br><br>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.<br><br>When this control is set to true, the channel separately processes these queries without interrupting the normal processing of events.<br><br>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.<br><br>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.<br><br>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.<br><br>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.<br><br>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.<br><br>While using this control, ensure the following:<br><br>◆ The value is not left empty. By default, it is populated with ***$***. This character is required for local variable expansion.<br>◆ The value should be a valid comma(,) separated list, otherwise you will encounter errors during policy evaluation.<br>◆ To escape all meta-characters, specify "\,$,^,.,?,*,+,[,],(,),\|" as a value.<br>◆ If a meta-character need not be escaped, remove that character from the value.<br>◆ To escape any meta character, specify the meta character followed by a back slash (\). |

# 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 Reporting Guide for Sentinel*.

| 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. |

# Driver Image/iManager Icon

Allows you to change the image associated with the driver in iManager. You can browse to and select a different image from the default image.

The image associated with a driver is used by the Identity Manager Overview plug-in when showing the graphical representation of your Identity Manager configuration. Although storing an image is optional, it makes the overview display more intuitive.

**NOTE:** The driver image is maintained when a driver configuration is exported.

# 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. |
| **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.<br><br>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 5, "Monitoring Driver Health," on page 19.

# 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 8, "Managing Driver Cache Files," on page 37.

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 7, "Managing Associations between Drivers and Objects," on page 31.

Designer does not include the Driver Inspector.

# B The Cache Flush Parameter

Identity Manager 4.5 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.