Novell Identity Manager Driver for SOAP

3.5.1

www.novell.com

IMPLEMENTATION GUIDE

September 28, 2007

Novell®

Legal Notices

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

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

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

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

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Online Documentation: To access the latest online documentation for this and other Novell products, see the Novell Documentation Web page (http://www.novell.com/documentation).

Novell Trademarks

For Novell trademarks, see the Novell Trademark and Service Mark list (http://www.novell.com/company/legal/trademarks/tmlist.html).

Third-Party Materials

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

Contents

About This Guide 9

1 Overview 11

1.1 What's New . . . 11
1.1.1 Driver Features . . . 11
1.1.2 Identity Manager Features . . . 11
1.2 Driver Concepts . . . 11
1.2.1 Data Management . . . 12
1.2.2 How the Driver Works . . . 13
1.2.3 Understanding Operation Data . . . 13
1.3 Key Driver Features . . . 14
1.3.1 Local Platforms . . . 14
1.3.2 Remote Platforms . . . 14
1.3.3 Role-Based Entitlements . . . 15
1.3.4 Password Synchronization Support . . . 15
1.3.5 Information Synchronized . . . 15

2 Installing the Driver 17

2.1 Driver Prerequisites . . . 17
2.2 Installing the Driver . . . 17

3 Upgrading the Driver 19

3.1 Upgrading the Driver in Designer . . . 19
3.2 Upgrading the Driver in iManager . . . 22

4 Using the Sample Driver Configurations 23

4.1 Creating a Driver Object by Using a Driver Configuration File . . . 23
4.1.1 Importing the Driver Configuration File in Designer . . . 23
4.1.2 Importing the Driver Configuration File in iManager . . . 24
4.1.3 Configuration Parameters . . . 25
4.2 Understanding the DSML Configuration . . . 28
4.3 Understanding the SPML Configuration . . . 29
4.4 Handling Modify Events on the Publisher Channel for Unassociated Objects . . . 29

5 Configuring the Driver 31

5.1 Configuring Driver Settings . . . 31
5.2 Configuring Subscriber Settings . . . 32
5.2.1 Configuring the Subscriber to Make HTTPS Connections to the Remote Web Service 32
5.2.2 Configuring the Subscriber to Use a Proxy . . . 32
5.3 Configuring Publisher Settings . . . 33
5.3.1 Configuring the Publisher to Receive HTTPS Connections . . . 33
5.4 Creating XSLT Style Sheets . . . 34
5.5 Operation Data . . . 34
5.5.1 Using Operation Data to Specify XML to Be Returned on the Result . . . 35

5.5.2 Using Operation Data to Override Default Subscriber Options 35

6 Activating the Driver 37

7 Managing the Driver 39

7.1 Starting, Stopping, or Restarting the Driver 39
7.1.1 Starting the Driver in Designer 39
7.1.2 Starting the Driver in iManager 39
7.1.3 Stopping the Driver in Designer 39
7.1.4 Stopping the Driver in iManager 39
7.1.5 Restarting the Driver in Designer 40
7.1.6 Restarting the Driver in iManager 40
7.2 Migrating and Resynchronizing Data 40
7.3 Using the DirXML Command Line Utility 40
7.4 Viewing Driver Versioning Information 41
7.4.1 Viewing a Hierarchical Display of Versioning Information 41
7.4.2 Viewing the Versioning Information as a Text File 43
7.4.3 Saving Versioning Information 45
7.5 Reassociating a Driver Set Object with a Server Object 46
7.6 Changing the Driver Configuration 46
7.7 Storing Driver Passwords Securely with Named Passwords 46
7.7.1 Using Designer to Configure Named Passwords 47
7.7.2 Using iManager to Configure Named Passwords 47
7.7.3 Using Named Passwords in Driver Policies 49
7.7.4 Using the DirXML Command Line Utility to Configure Named Passwords 49
7.8 Adding a Driver Heartbeat 53

8 Synchronizing Objects 55

8.1 What Is Synchronization? 55
8.2 When Is Synchronization Done? 55
8.3 How Does the Metadirectory Engine Decide Which Object to Synchronize? 56
8.4 How Does Synchronization Work? 57
8.4.1 Scenario One 57
8.4.2 Scenario Two 59
8.4.3 Scenario Three 60

9 Troubleshooting the Driver 63

9.1 Driver Shim Errors 63
9.2 Java Customization Errors 66
9.3 Troubleshooting Driver Processes 67
9.3.1 Viewing Driver Processes 67

10 Backing Up the Driver 75

10.1 Exporting the Driver in Designer 75
10.2 Exporting the Driver in iManager 75

11 Security: Best Practices 77

A Using Java Extensions 79

A.1 Overview . . . 79
A.2 Creating and Configuring Java Extensions . . . 80

B DirXML Command Line Utility 83

B.1 Interactive Mode . . . 83
B.2 Command Line Mode . . . 92

C Properties of the Driver 97

C.1 Driver Configuration . . . 97
C.1.1 Driver Module . . . 98
C.1.2 Driver Object Password . . . 98
C.1.3 Authentication . . . 99
C.1.4 Startup Option . . . 100
C.1.5 Driver Parameters . . . 101
C.2 Global Configuration Values . . . 104
C.3 Named Passwords . . . 105
C.4 Engine Control Values . . . 106
C.5 Log Level . . . 108
C.6 Driver Image . . . 109
C.7 Security Equals . . . 109
C.8 Filter . . . 110
C.9 Edit Filter XML . . . 110
C.10 Misc . . . 111
C.11 Excluded Users . . . 111
C.12 Driver Manifest . . . 112
C.13 Driver Inspector . . . 112
C.14 Driver Cache Inspector . . . 113
C.15 Inspector . . . 113
C.16 Server Variables . . . 114

About This Guide

This guide explains how to install and configure the Identity Manager Driver 1.0 for SOAP (also called the SOAP driver).

- Chapter 1, “Overview,” on page 11
- Chapter 2, “Installing the Driver,” on page 17
- Chapter 3, “Upgrading the Driver,” on page 19
- Chapter 4, “Using the Sample Driver Configurations,” on page 23
- Chapter 5, “Configuring the Driver,” on page 31
- Chapter 6, “Activating the Driver,” on page 37
- Chapter 7, “Managing the Driver,” on page 39
- Chapter 8, “Synchronizing Objects,” on page 55
- Chapter 9, “Troubleshooting the Driver,” on page 63
- Chapter 10, “Backing Up the Driver,” on page 75
- Chapter 11, “Security: Best Practices,” on page 77
- Appendix A, “Using Java Extensions,” on page 79
- Appendix B, “DirXML Command Line Utility,” on page 83
- Appendix C, “Properties of the Driver,” on page 97

Audience

This guide is intended for administrators implementing Identity Manager, application server developers, Web services administrators, and consultants. You should also have an understanding of DSML/SPML, SOAP, and HTML.

Feedback

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

Documentation Updates

For the most recent version of this document, see Identity Manager Driver for SOAP in the Drivers Documentation Web site (http://www.novell.com/documentation/idm35drivers/index.html).

Additional Documentation

For information on Identity Manager, see the Identity Manager Documentation Web site (http://www.novell.com/documentation/idm35).

Documentation Conventions

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

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

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

Overview

1

SOAP (Simple Object Access Protocol) is an XML-based protocol used for Internet communication between different applications and operating systems.

The SOAP driver uses a combination of language and protocols to enable identity provisioning and data synchronization between an Identity Vault with Identity Manager and an HTTP-enabled application, such as a SOAP-enabled Web service.

The driver isn't targeted to a specific Web service. The driver is a generic shim that simply handles the HTTP transport of data between the Identity Vault and a Web service. For this driver, a Web service is defined as an application that uses XML and HTTP as the transport protocol. The application can also use SOAP to encode the messages.

This section provides the following information on the Identity Manager Driver for SOAP:

- Section 1.1, "What's New," on page 11
- Section 1.2, "Driver Concepts," on page 11
- Section 1.3, "Key Driver Features," on page 14

1.1 What's New

In this section:

- Section 1.1.1, "Driver Features," on page 11
- Section 1.1.2, "Identity Manager Features," on page 11

1.1.1 Driver Features

There are no new features for the SOAP driver with the release of Identity Manager 3.5.1.

1.1.2 Identity Manager Features

For information about the new features in Identity Manager, see "What's New in Identity Manager 3.5?" in the *Identity Manager 3.5.1 Installation Guide*.

1.2 Driver Concepts

This section contains the following information:

- "Data Management" on page 12
- "How the Driver Works" on page 13

1.2.1 Data Management

The driver uses various Internet protocols and languages to exchange data between Identity Manager and a Web service.

- "SOAP" on page 12
- "SPML and DSML" on page 12
- "XML" on page 12
- "HTTP" on page 13

SOAP

SOAP (Simple Object Access Protocol) is an XML-based protocol for exchanging messages. It defines the message exchange but not the message content. The driver supports SOAP 1.1.

SOAP documents are organized into three elements:

- **Envelope:** The root XML node.
- **Header:** Provides context knowledge such as a transaction ID and security information.
- **Body:** The method-specific information.

SOAP follows the HTTP request/response message model, which provides SOAP request parameters in an HTTP request and SOAP response parameters in an HTTP response.

SPML and DSML

The SOAP driver includes sample configurations for two protocols: SPML 1.0 and DSML 2.0.

- **SPML 1.0:** Service Provisioning Markup Language is an XML-based provisioning request and response protocol. A client issues an SPML request to a server. The request describes the operation to be performed at a given service point. The service point performs the necessary operations to implement the requested service. After completing the operation, the service point returns an SPML response to the client detailing any results or errors pertinent to that request.

  The driver supports SPML 1.0. SPML binds with SOAP 1.1 and uses HTTP and HTTPS 1.1 as the transport.
- **DSML 2.0:** Directory Services Markup Language represents directory structural information, directory queries and updates, and the results of these operations as XML documents.

DSML binds with SOAP 1.1 and uses HTTP and HTTPS 1.1 as the transport.

For more information about the sample SPML and DSML configurations included with the driver, see "Using the Sample Driver Configurations" on page 23.

XML

XML (Extensible Markup Language) is a generic subset of Standard Generalized Markup Language (SGML) that allows for exchange of structured data on the Internet.

HTTP

HTTP is a protocol used to request and transmit data over the Internet or other computer network. The protocol works well in an Internet infrastructure and with firewalls.

HTTP is a stateless request/response system because the connection is usually maintained only for the immediate request. The client establishes a TCP connection with the server and sends it a request command. The server then sends back its response.

1.2.2 How the Driver Works

The following diagram illustrates the data flow between Identity Manager and a Web service:

Figure 1-1 *SOAP Driver Data Flow*

The Identity Manager engine uses XDS, a specialized form of XML, to represent events in the Identity Vault. Identity Manager passes the XDS to the driver policy, which can consist of basic policies, DirXML® Script, and XSLT style sheets.

The driver policy translates the XDS to XML, such as SOAP, on the Subscriber channel. On the Publisher channel, the driver policy translates other forms of XML, such as SOAP, into XDS.

The driver shim receives the XML from the driver policy. The driver shim uses HTTP to communicate with the Web service. Generally the handoff between the driver shim and the application is serialized XML.

For example, suppose the driver is using the DSML sample configuration to talk to a DSML server that is configured only as a Subscriber. When an event occurs in the Identity Vault, Identity Manager creates an XDS command to represent that event. Identity Manager passes the XDS command to the driver policy.

The driver policy transforms that XDS command with an output transformation style sheet. The XSLT style sheet converts the XDS to a SOAP envelope containing DSML. That SOAP envelope is handed to the driver shim. The driver shim converts the SOAP envelope into an array of bytes, makes the appropriate HTTP connection, and performs an HTTP POST operation to submit the data to the Web service.

The Web service or application processes the request, and returns a SOAP response to the driver shim. The shim receives the response as an array of bytes, and converts it to an XML document before passing it back to the driver policies. The input transformation style sheet processes the response, converting it into appropriate XDS that is reported back to the Identity Manager engine.

1.2.3 Understanding Operation Data

The driver shim applies special handling to Subscriber commands based on an XML element embedded in the command, which appears in the driver shim as `<operation-data>`. The `<operation-data>` element has two purposes. First, it can be used to match commands with the responses they generate, which can be useful for creating associations. Second, it can be used to override default Subscriber channel connection attributes.

The <operation-data> element is added to the command from one of the Subscriber channel policies. The driver shim removes the <operation-data> element from the command before it is sent to the application, and restores the <operation-data> element to the resulting response.

By default, when the <operation-data> element is restored on the response, it is appended as a child element of the root node. This can be overridden by providing one or more parent-node-*n* attributes to the <operation-data> element, where *n* is a number beginning with 1 that is incremented for each parent specifier you want to provide. The driver shim examines the operation data node, looking for parent-node-*n* attributes. If attributes are found, each is tried in turn and if the named node exists, the node is used as the parent for the operation data on the response.

To see how the <operation-data> element works with the style sheets, see Section 5.5, "Operation Data," on page 34.

1.3 Key Driver Features

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

- Section 1.3.1, "Local Platforms," on page 14
- Section 1.3.2, "Remote Platforms," on page 14
- Section 1.3.3, "Role-Based Entitlements," on page 15
- Section 1.3.4, "Password Synchronization Support," on page 15
- Section 1.3.5, "Information Synchronized," on page 15

1.3.1 Local Platforms

The SOAP driver can be installed locally on the following platforms:

- NetWare® 6 or 6.5 with the latest Support Pack
- Novell® Open Enterprise Server with the latest Support Pack
- Windows* NT*, 2000, or 2003 with the latest Service Patch
- Linux Red Hat* AS, ES 2.1, or AS 3.0
- SUSE® Linux Enterprise Server 8, 9 (including SP1), or 10
- Solaris* 8 or 9
- AIX* 5.2L

1.3.2 Remote Platforms

The SOAP driver can use the Remote Loader service. The Remote Loader service for the SOAP driver can be installed on the following platforms:

- Novell Open Enterprise Server with the latest Support Pack
- Windows NT, 2000, or 2003 with the latest Service Patch
- Linux Red Hat AS, ES 2.1, or AS 3.0
- SUSE LINUX Enterprise Server 8, 9 (including SP1), or 10
- Solaris 8 or 9
- AIX 5.2L

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

1.3.3 Role-Based Entitlements

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

1.3.4 Password Synchronization Support

The example configuration files for the SOAP driver are capable of synchronizing passwords.

1.3.5 Information Synchronized

Unlike most other drivers, the SOAP driver synchronizes protocols instead of objects. It synchronizes the SPML 1.0 and DSML 2.0 protocols. The driver contains the following features:

- HTTP transport of data between the Identity Vault and a Web service
- Example configurations for SPML and DSML
- Customization of HTTP Request-Header fields

  By default, a basic authorization request header with an ID and password is provided for the Subscriber channel. For more information, see Section 4.1, "Creating a Driver Object by Using a Driver Configuration File," on page 23.
- SSL connections using the HTTPS protocol
- Subscriber HTTP and HTTPS proxy servers
- Definition and selection of multiple Subscriber connections in the policy at runtime
- Potential to act as an HTTP or HTTPS listener for incoming connections on the publisher channel
- Potential extensibility using customized Java* code

  For more information, see Appendix A, "Using Java Extensions," on page 79.

Installing the Driver

2

The section contains the following information about installing the driver:

- Section 2.1, “Driver Prerequisites,” on page 17
- Section 2.2, “Installing the Driver,” on page 17

2.1 Driver Prerequisites

- ❑ One of the following operating systems:
  - NetWare® 6 or 6.5 with the latest Support Pack
  - Novell® Open Enterprise Server with the latest Support Pack
  - Windows NT, 2000, or 2003 with the latest Service Patch
  - Linux Red Hat AS, ES 2.1, or AS 3.0
  - SUSE® Linux Enterprise Server 8, 9 (including SP1), or 10
  - Solaris 8 or 9
  - AIX 5.2L
- ❑ Novell eDirectory™ 8.7.3 with the latest Support Pack, or Novell eDirectory 8.8 with the latest Support Pack
- ❑ Novell Identity Manager 3.5.1
- ❑ Novell iManager 2.6 or higher

2.2 Installing the Driver

You install the driver as part of the Novell Identity Manager 3.5.1 installation program. For installation instructions, refer to the “Installing Identity Manager” in the*Identity Manager 3.5.1 Installation Guide*. If you are upgrading the driver, see Chapter 3, “Upgrading the Driver,” on page 19.

Importing the driver configuration creates the driver object. After you have imported the configuration, you can use iManager to configure and manage the driver. See Chapter 4, “Using the Sample Driver Configurations,” on page 23 for instructions on how to configure the driver.

Upgrading the Driver

3

If you have been using a previous version of the driver, follow these instructions instead of the ones in Chapter 2, “Installing the Driver,” on page 17.

Identity Manager 3.5 contains a new architecture for how policies reference one another. To take advantage of this new architecture, the driver configuration file provided for SOAP must be upgraded. For more information on the new architecture, see “Upgrading Identity Manager Policies” in the *Understanding Policies for Identity Manager 3.5.1*. You can upgrade the driver in Designer or iManager.

If you are upgrading from Identity Manager 3.5.0 to Identity Manager 3.5.1, the following information does not apply.

- Section 3.1, “Upgrading the Driver in Designer,” on page 19
- Section 3.2, “Upgrading the Driver in iManager,” on page 22

3.1 Upgrading the Driver in Designer

1. Make sure you have updated your driver with all the patches for the version you are currently running.

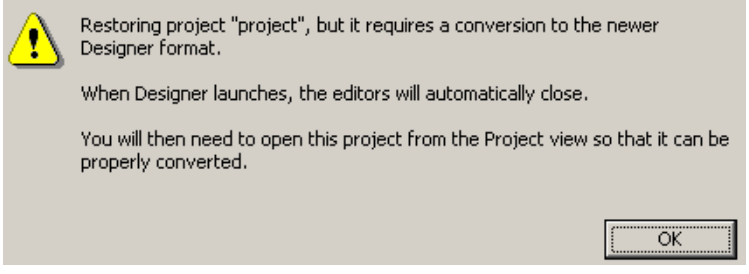
   We recommend this step for all drivers, to help minimize upgrade issues.

2. Back up the driver. See Chapter 10, “Backing Up the Driver,” on page 75 for instruction on how to back up the driver.

3. Install Designer version 2.0 or above, then launch Designer.

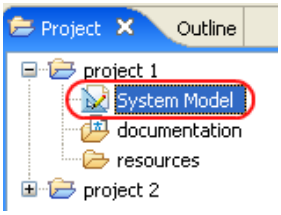
   If you had a project open in Designer when you upgraded Designer, proceed to Step 4. If you didn’t have a project open in Designer when you upgraded Designer, skip to Step 5.

4. If you had a project open when upgrading Designer, the following warning message is displayed. Read the warning message, then click *OK*.

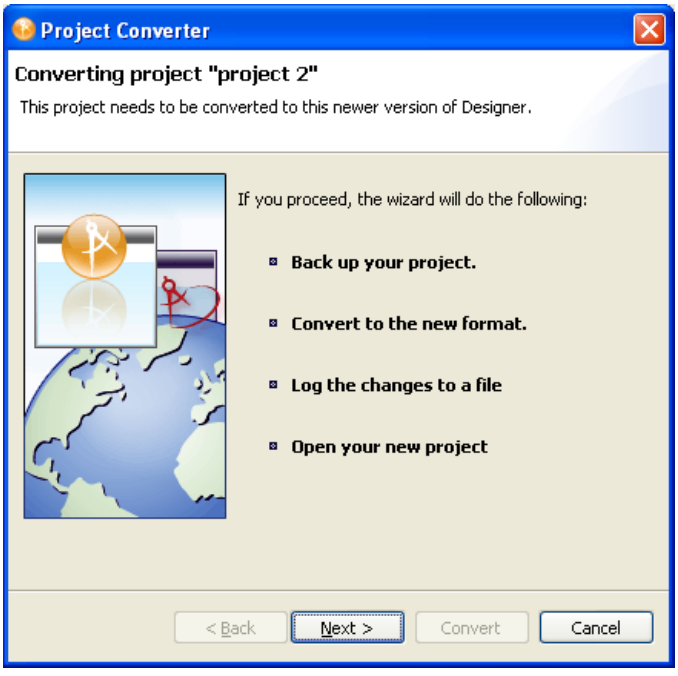


   Designer closes the project to preform the upgrade.

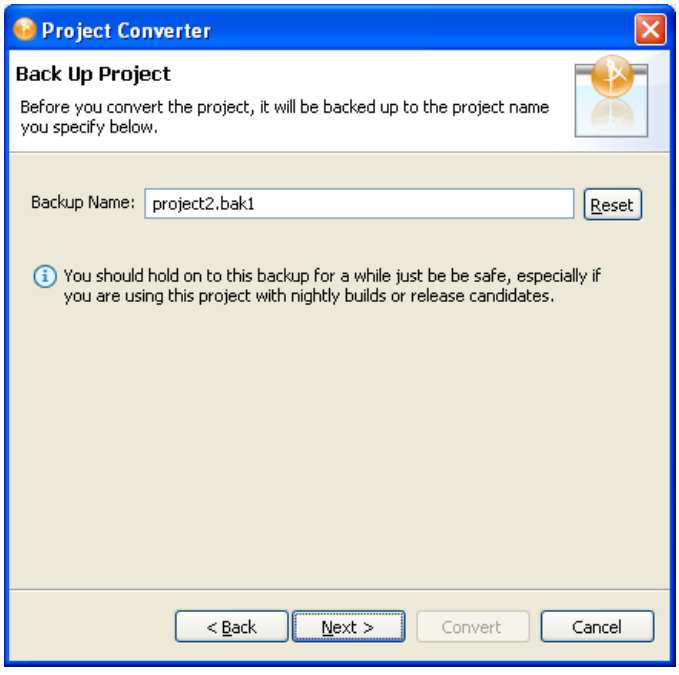
5 In the Project view, double-click *System Model* to open and convert the project.



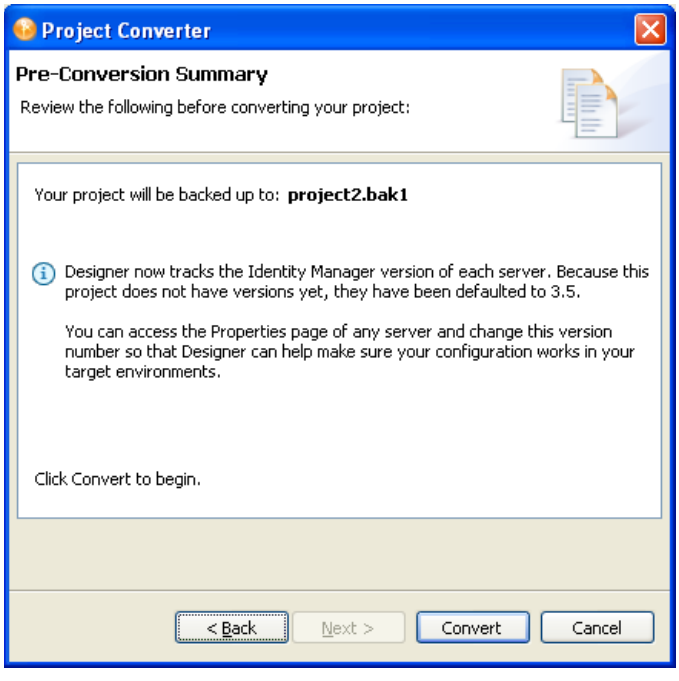
6 Read the Project Converter message explaining that the project is backed up, converted to the new format, changes logged to a file, and the new project is opened, then click *Next*.



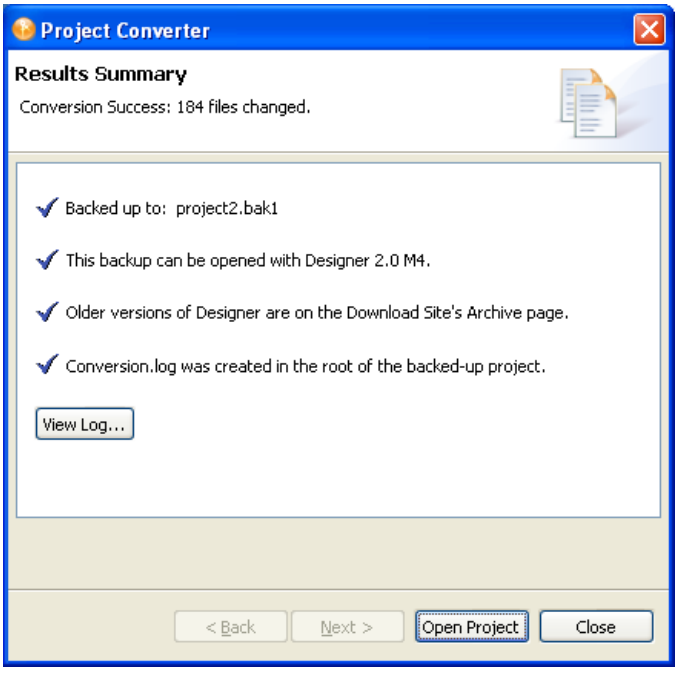
7 Specify the name of the backup project name, then click *Next*.



8 Read the project conversion summary, then click *Convert*.



9 Read the project conversion result summary, then click *Open Project*.



If you want to view the log file that is generated, click *View Log*.

3.2 Upgrading the Driver in iManager

1 Make sure you have updated your driver with all the patches for the version you are currently running.

We recommend this step for all drivers, to help minimize upgrade issues.

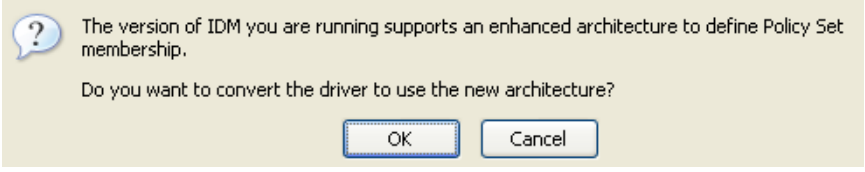
2 Back up the driver. See Chapter 10, “Backing Up the Driver,” on page 75 for instruction on how to back up the driver.

3 Verify that Identity Manager 3.5.1 has be installed and you have the current plug-ins installed, then launch iManager.

4 Click *Identity Manager* > *Identity Manager Overview*.

5 Click *Search* to find the Driver Set object, then click the driver you want to upgrade.

6 Read the message that is displayed, then click *OK*.



7 If there is more than one driver to upgrade, repeat Step 2 through Step 6.

Using the Sample Driver Configurations

4

The Identity Manager Driver for SOAP includes two sample configurations that you can use as a starting point for creating the Driver object.

This section covers the following topics:

- Section 4.1, "Creating a Driver Object by Using a Driver Configuration File," on page 23
- Section 4.3, "Understanding the SPML Configuration," on page 29
- Section 4.2, "Understanding the DSML Configuration," on page 28

4.1 Creating a Driver Object by Using a Driver Configuration File

The SOAP driver comes with two configuration files that can be used to create a Driver object:

- SOAP-SPML-IDM3_5_0-V1.xml: The Service Provisioning Markup Language (SPML) configuration file
- SOAP-DSML-IDM3_5_0-V1.xml: The Directory Services Markup Language (DSML) configuration file

For more information about the sample files, see Section 4.3, "Understanding the SPML Configuration," on page 29 and Section 4.2, "Understanding the DSML Configuration," on page 28.

4.1.1 Importing the Driver Configuration File in Designer

Designer allows you to import the driver configuration files for the SOAP driver. These files create and configure the objects and policies needed to make the driver work properly. The following instructions explain how to create the driver and import the driver's configuration.

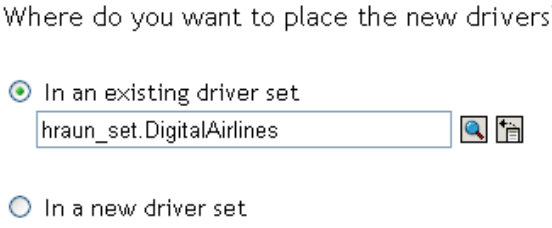
There are many different ways of importing the driver configuration file. This procedure only documents one way.

1. Open a project in Designer. In the Modeler, right-click the Driver Set object, then select *New* > *Driver*.
2. From the drop-down list, select *SOAP DSML* or *SOAP SPML,* then click *Run*.
3. Configure the driver by filling in the fields. Specify information specific to your environment. For information on the settings, see Table 4-1 on page 25 and Table 4-2 on page 27.
4. After specifying parameters, click *Finish* to import the driver.
5. After the driver is imported, customize and test the driver.
6. After the driver is fully tested, deploy the driver into the Identity Vault. See "Deploying a Driver to an Identity Vault" in *Designer 2.1 for Identity Manager 3.5.1*.

4.1.2 Importing the Driver Configuration File in iManager

The SOAP preconfiguration files are an example configuration file. You installed this file when you installed the Identity Manager Web components on an iManager server. Think of the preconfiguration file as a template that you import and customize or configure for your environment.

1. In iManager, select *Identity Manager Utilities* > *Import Drivers*.
2. Select a driver set, then click *Next*.



   If you place this driver in a new driver set, you must specify a driver set name, context, and associated server.
3. Select how you want the driver configurations sorted:
   - All configurations
   - Identity Manager 3.5 configurations
   - Identity Manager 3.0 configurations
   - Configurations not associated with an IDM version
4. Select *SOAP DSML* or *SOAP SPML*, then click *Next*.

5. Configure the driver by filling in the configuration parameters, then click *Next*. For information on the settings, see Table 4-1 on page 25 and Table 4-2 on page 27.
6. Define security equivalences using a user object that has the rights that the driver needs to have on the server

   The Admin user object is most often used for this task. However, you might want to create a DriversUser (for example) and assign security equivalence to that user. Whatever rights that the driver needs to have on the server, the DriversUser object must have the same security rights.
7. Identify all objects that represent administrative roles and exclude them from replication.

   Exclude the security-equivalence object (for example, DriversUser) that you specified in Step 6. If you delete the security-equivalence object, you have removed the rights from the driver. Therefore, the driver can't make changes to Identity Manager.
8. Click *Finish*.
9. Configure additional settings for the driver.

   For more information, see "Configuring the Driver" on page 31.

4.1.3 Configuration Parameters

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

NOTE: The parameters are presented on multiple screens and some parameters are only displayed if the answer to a previous prompt requires more information to properly configure the policy.

Table 4-1 *Configuration Parameters for the SOAP DSML Driver*

Field	Description
Driver name	Specify the name of the driver object in Identity Manager.
Configure Data Flow	Specify the driver channels you want to be active. *eDirectory to DSML*: Sends Identity Vault events to the application. *DSML to eDirectory*: Receives events from the application. *Bi-Directional*: Activates both the eDirectory™ and the DSML channels.
<nds>, <input>, <output> element handling	Select one of the following: *Remove/Add Elements*: The driver shim removes and adds the required XML elements of nds, input, and output. These required elements are removed from XML documents sent to the application and are added to XML documents received from the application before sending the document to the Metadirectory engine. This is the preferred option for the SOAP driver. *Pass Elements Through*: Turns off element handling. The required XML elements of nds, input, and output aren't added or removed to XML documents as necessary.
Driver is Local/Remote	Select one of the following: *Local*: Runs the driver shim from the server holding the driver set. *Remote*: Runs the driver from a remote server using the Remote Loader. If you specify this option, click *Next*, then specify Remote Loader configuration information. For more information, see "Deciding Whether to Use the Remote Loader" in the *Novell Identity Manager 3.5.1 Administration Guide*.

Field	Description
URL of the remote DSML server: (Conditional) Subscriber Channel fields **NOTE:** These fields are displayed only if you select *eDirectory to DSML* or *Bi-Directional* in the *Configure Data Flow* field.	Specify the *URL of the remote DSML server* and the port number that the server listens on. For example: http://137.66.10.13:18180/soap The server is a software component that listens for, processes, and returns the results for valid DSML requests. **TIP:** If you configure the driver to use SSL, the URL must begin with https rather than http.
Authentication ID (Conditional) Subscriber Channel fields	If the remote server requires an *Authentication ID*, specify it in the field. Otherwise, leave the field empty.
Authentication Password (Conditional) Subscriber Channel fields	Specify the *Authentication Password* for the remote server if you specified an *Authentication ID* above. Otherwise, leave these fields empty.
Listening IP address and port (Conditional) Publisher Channel fields **NOTE:** These fields are displayed only if you select *DSML to eDirectory* or *Bi-Directional* in the *Configure Data Flow* field.	Specify the IP address of the server where the SOAP driver is installed and the port number that this driver listens on. You can specify 127.0.0.1 if there is only one network card installed in the server. Choose an unused port number on your server, for example, 127.0.0.1:18180. The driver listens on this address for requests, processes the requests, and returns a result.
Authentication ID (Conditional) Publisher Channel fields	Specify the *Authentication ID* of the remote DSML server to validate incoming requests. If the remote server does not send an *Authentication ID*, leave this field empty.
Authentication Password (Conditional) Publisher Channel fields	Specify the *Authentication Password* of the remote server to validate incoming requests, if you specified an *Authentication ID* above. Otherwise, leave these fields empty.
Remote Host Name and Port (Conditional) Remote Loader fields **NOTE:** These fields are displayed only if you select *Remote* in the *Driver is Local/Remote* field.	Specify the host name or IP address of the server running the remote loader server and port. Example: 137.66.10.13:8090 Port 8090 is the default port the Remote Loader service listens on.
Driver password (Conditional) Remote Loader fields	The driver password is used by the Remote Loader to authenticate itself to the Identity Manager server. It must be the same password that is specified in the driver object password on the Remote Loader server.
Remote password (Conditional) Remote Loader fields	The remote password is used to control access to the Remote Loader. It must be the same password that is specified as the Remote Loader password on the Remote Loader server.

Table 4-2 *Configuration Parameters for the SOAP SPML Driver*

Field	Description
Driver name	Specify the name of the driver object in Identity Manager.
Configure Data Flow	Specify the driver channels you want to be active. *eDirectory to SPML:* Sends Identity Vault events to the application. *SPML to eDirectory*: Receives events from the application. *Bi-Directional*: Activates both the eDirectory and the SPML channels.
<nds>, <input>, <output> element handling	Select one of the following: *Remove/Add Elements*: The driver shim removes and adds the required XML elements of nds, input, and output. These required elements are removed from XML documents sent to the application and are added to XML documents received from the application before sending the document to the Metadirectory (Identity Manager) engine. This is the preferred option for the SOAP Driver. *Pass Elements Through*: Turns off element handling. The required XML elements of nds, input, and output aren't added or removed to XML documents as necessary.
Driver is Local/Remote	Select one of the following: *Local*: Runs the driver shim from the server holding the driver set. *Remote*: Runs the driver from a remote server using the Remote Loader. If you specify this option, click *Next*, then specify Remote Loader configuration information. For more information, see "Deciding Whether to Use the Remote Loader" in the *Novell Identity Manager 3.5.1 Administration Guide*.
URL of the remote SPML Provisioning Service Point: (Conditional) Subscriber Channel fields **NOTE:** These fields are displayed only if you select *eDirectory to SPML* or *Bi-Directional* in the *Configure Data Flow* field.	Specify the URL of the remote SPML Provisioning Service Point (PSP). For example: http://137.66.10.13:18180/soap A PSP is a software component that listens for, processes, and returns the results for valid SPML requests. **TIP:** If you configure the driver to use SSL, the URL must begin with https rather than http.
Authentication ID (Conditional) Subscriber Channel fields	Specify the authentication ID of the remote SPML PSP. If the remote SPML PSP requires an authentication ID. Otherwise, leave the field empty.
Authentication Password (Conditional) Subscriber Channel fields	Specify the authentication password for the remote SPML PSP to validate incoming requests, if you specified an authentication ID above. Otherwise, leave this field empty.

Field	Description
Listening IP address and port (Conditional) Publisher Channel fields **NOTE:** These fields are displayed only if you select *SPML to eDirectory* or *Bi-Directional* in the *Configure Data Flow* field.	Specify the IP address of the server where the driver is installed and the port number that this driver listens on as a PSP. You might specify 127.0.0.1 if there is only one network card installed in the server. Choose an unused port number on your server. Example: 127.0.0.1:18180 The driver listens on this address for the SPML requests, processes them, and returns a result.
Authentication ID (Conditional) Publisher Channel fields	Specify the authentication ID to validate incoming SPML requests.
Authentication Password (Conditional) Publisher Channel fields	Specify the authentication password to validate incoming SPML requests.
Remote Host Name and Port (Conditional) Remote Loader fields **NOTE:** These fields are displayed only if you select *Remote* in the *Driver is Local/Remote* field.	Enter the hostname or IP address of the server running the Remote Loader server and port. Example: 137.66.10.13:8090 Port 8090 is the default port the Remote Loader service listens on.
Driver password (Conditional) Remote Loader fields	The driver password is used by the Remote Loader to authenticate itself to the Identity Manager server. It must be the same password that is specified in the driver object password on the Remote Loader server.
Remote password (Conditional) Remote Loader fields	The remote password is used to control access to the Remote Loader. It must be the same password that is specified as the Remote Loader password on the Remote Loader server.

4.2 Understanding the DSML Configuration

The sample DSML configuration uses DSML 2.0 and binds with SOAP 1.1 using HTTP or HTTPS 1.1 as the transport. All data transformation and processing is done in policies and style sheets.

The sample DSML import file does the following:

- Shows a simple configuration for pairing with the Identity Vault DSML implementation.
- Provides XDS-to-DSML and DSML-to-XDS conversions in policies.
- Handles Users, Groups and Organizational Units.

  Other objects can be processed through policy and style sheet customization.
- Supports string, structured, and distinguished name (DN) attribute types.

  There are two examples of handling attributes with other data types. The Postal Address attribute shows how structured attributes can be handled. The Member attribute shows how a DN attribute can be handled. Other attribute data types can be handled through policy and style sheet customization.
- Handles a subset of the query operations.

Specific query operations can be handled through policy and style sheet customizations.

- Supports password set operation.

  Password synchronization might be possible through policy and style sheet customization.
- The Subscriber channel uses the destination DN for the association key.
- The Publisher channel uses the application-provided DN for the association key.

4.3 Understanding the SPML Configuration

The sample SPML configuration uses SPML 1.0 and binds with SOAP 1.1 using HTTP or HTTPS 1.1 as the transport. All data transformation and processing is done in policies and style sheets.

The sample SPML import file does the following:

- Provides generic SPML functionality.

  The import file doesn't pair with a specific SPML application.
- Provides XDS-to-SPML and SPML-to-XDS conversions in policies.
- Handles Users, Groups, and Organizational Units

  Other objects can be handled through policy and style sheet customization.
- Handles a single value per attribute.

  Multiple values for an attribute can be handled through policy and style sheet customization.
- Handles a subset of the query operations.

  The configuration handles all queries as SPML scope = "subtree" and uses the entry and subordinate scope concepts. Specific query operations can be handled through policy and style sheet customization.
- Supports string, structured, and distinguished name (DN) attribute types.
- Supports password set operation.

  Password synchronization might be possible through policy and style sheet customization.
- Handles the single (non-batch) operations of execution=synchronous and processing=sequential.

  Batch requests can be supported through policy and style sheet customization.
- Doesn't handle `<addResponse><attributes>` or `<modifyResponse><modifications>`.
- The Subscriber channel uses the application-returned Identifier value for the association key.
- The Publisher channel uses the DN for the association key and returns the association key as the Identifier value.

4.4 Handling Modify Events on the Publisher Channel for Unassociated Objects

The Publisher channel of the HTTP/SOAP driver has certain limitations that allow it to listen only for Change events. It has no way to query for additional information or to poll the HTTP/SOAP source. Therefore, Modify events received on the Publisher channel for unassociated objects (or an object that was not created by the same instance of the driver) almost always fail (return an error). The reason for this is that the driver and the Metadirectory engine cannot successful change an

unassociated Modify event into an Add command without the ability to send a query to the HTTP/SOAP source. Because the SOAP driver has no mechanism to query back to the source, it returns an error stating that query is not implemented.

There is no general solution for this limitation. Therefore, the sample configurations for DSML and SPML both return an error when this condition occurs. If, in a specific driver deployment, it becomes necessary to apply an association on an object, and the possibility of inconsistent information in that newly associated object is satisfactory, this can be achieved in policy by setting the Destination DN in the Modify event and creating your own set-association event. This allows the modification to occur on the existing object, even when not previously associated.

Configuring the Driver

5

After you create the Driver object using one of the sample files, you need to configure the Identity Manager Driver for SOAP. This section contains the following information about configuring the driver:

- Section 5.1, “Configuring Driver Settings,” on page 31
- Section 5.2, “Configuring Subscriber Settings,” on page 32
- Section 5.3, “Configuring Publisher Settings,” on page 33
- Section 5.4, “Creating XSLT Style Sheets,” on page 34

5.1 Configuring Driver Settings

1. In iManager, click *Identity Manager* > *Identity Manager Overview*.
2. Locate the driver set that contains the SOAP driver, then click the driver’s icon.
3. From the *Identity Manager Driver Overview*, click the SOAP driver object, which displays the driver configurations.
4. Specify driver module information:
   - **4a** In the *Driver Module* section, select *Java*.
   - **4b** In the *Name* field, specify the following SOAP driver Java class name:

     ```
     com.novell.nds.dirxml.driver.soap.SOAPDriver
     ```
5. Specify driver object password information:
   - **5a** Scroll to the *Driver Object Password* section, then click *Set password*.
   - **5b** In the fields, enter and re-enter the driver object password.
6. Specify authentication information:
   - **6a** Scroll to the Authentication section.
   - **6b** Specify the *Authentication ID*.
   - **6c** Specify the *Authentication context*.
   - **6d** Specify *Remote loader connection parameters*.
   - **6e** Specify the *Driver cache limit* (kilobytes).
   - **6f** Specify *Application password* by clicking *Set password*.
   - **6g** Enter and re-enter the *Application password*.
7. Specify startup information:
   - **7a** Scroll to the *Startup* section.
   - **7b** Select one of the following:
     - *Auto Start*: The driver starts automatically when eDirectory™ starts.
     - *Manual*: The driver must be started manually using iManager.
     - *Disabled*: The driver will not run.
8. Specify the driver settings as described in Section C.1, “Driver Configuration,” on page 97.

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

5.2 Configuring Subscriber Settings

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

2 Locate the driver set containing the SOAP driver, then click the driver's icon to display the Identity Manager Driver Overview page.

3 On the Identity Manager Driver Overview page, click the driver's icon again, then scroll to *Subscriber Settings*.

4 Specify the Subscriber settings as described in Section C.1, "Driver Configuration," on page 97.

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

5.2.1 Configuring the Subscriber to Make HTTPS Connections to the Remote Web Service

If the remote Web service you are accessing allows HTTPS connections, you can configure the Subscriber channel to take advantage of this capability. You need a trust store containing a certificate issued by the certificate authority that signed the server's certificate. See Section 5.3.1, "Configuring the Publisher to Receive HTTPS Connections," on page 33 for an example.

Import this certificate into a trust store using Java's keytool. For more information on keytool, see Keytool - Key and Certificate Management Tool (http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/keytool.html).

1 Import the certificate into your trust store or create a new trust store by entering the following command at the command prompt:

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

For example:

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

2 Configure the Subscriber channel to use the trust store you created in Step 1.

2a In iManager, click *Identity Manager* > *Identity Manager Overview*.

2b Locate the driver set containing the SOAP driver, then click the driver's icon to display the Identity Manager Driver Overview page.

2c On the Identity Manager Driver Overview page, click the driver's icon again, then scroll to *Subscriber Settings*.

2d In the *Truststore File* setting, specify the path to the trust store you created in Step 1.

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

5.2.2 Configuring the Subscriber to Use a Proxy

You can configure the subscriber to use an HTTP or HTTPS proxy server.

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

2 Locate the driver set containing the SOAP driver, then click the driver's icon to display the Identity Manager Driver Overview page.

3 On the Identity Manager Driver Overview page, click the driver's icon again, then scroll to *Subscriber Settings*.

4 In the *Proxy Host and Port* field, specify the host and port of the proxy using the following format:

```
host:port
```

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

5.3 Configuring Publisher Settings

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

2 Locate the driver set containing the SOAP driver, then click the driver's icon to display the Identity Manager Driver Overview page.

3 On the Identity Manager Driver Overview page, click the driver's icon again, then scroll to *Publisher Settings*.

4 Specify the Publisher settings as described in Section C.1, "Driver Configuration," on page 97.

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

5.3.1 Configuring the Publisher to Receive HTTPS Connections

1 Create a server certificate in iManager.

1a Click *Novell Certificate Server* > *Create Server Certificate*.

1b Browse to and select the server object where the SOAP driver is installed.

1c Specify a certificate nickname.

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

1e Click *Finish*, then click *Close*.

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

2a Click *eDirectory Administration* > *Modify Object*.

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

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

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

2d Click *Export*.

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

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

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

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

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

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

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

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

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

For more information on keytool, see Keytool - Key and Certificate Management Tool (http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/keytool.html).

3b Type the following command at a command prompt:

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

For example:

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

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

4a In iManager, click *Identity Manager* > *Identity Manager Overview*.

4b Locate the driver set containing the SOAP driver, then click the driver's icon to display the Identity Manager Driver Overview page.

4c In the Identity Manager Driver Overview page, click the driver's icon again, then scroll to *Publisher Settings*.

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

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

5.4 Creating XSLT Style Sheets

To enable the SOAP driver to work with any setup other than the default configuration for DSML or SPML, you need to create XSLT style sheets.The application-specific protocol handling is done in Input Transformation and Output Transformation style sheets.

For detailed information on writing style sheets to handle other document types, refer to the sample style sheets that come with this driver. For more information on style sheets see "Defining Policies by Using XSLT Style Sheets" in the *Understanding Policies for Identity Manager 3.5.1*.

5.5 Operation Data

The driver shim applies special handling to Subscriber commands based on the `<operation-data>` element. On the Subscriber channel, the `<operation-data>` element can be added to a command for two purposes.

1. Specify XML data that you want included with the command result. In this way you can match commands with the responses they generate, which is useful for creating associations.
2. Override default Subscriber options on a per-command basis.

As discussed in Chapter 1, "Overview," on page 11, the `<operation-data>` element is added to the command from one of the Subscriber channel policies. The driver shim removes the operation data from the command before it is sent to the application, and restores the `<operation-data>`

element (and all child elements) to the resulting response. If needed, rules and style sheets can then access the operation-data element on the result.

- Section 5.5.1, "Using Operation Data to Specify XML to Be Returned on the Result," on page 35
- Section 5.5.2, "Using Operation Data to Override Default Subscriber Options," on page 35

5.5.1 Using Operation Data to Specify XML to Be Returned on the Result

The sample configurations for the SOAP driver use the `<operation-data>` element to keep track of identifying information for a command, so the result can be recognized and associations can be properly assigned. Check these samples for details of how the `<operation-data>` element is used.

When the `<operation-data>` element is restored on the response, it appended as a child element of the root node. You can override this by providing one or more parent-node-*n* attributes to the `<operation-data>` element where *n* is a number beginning with 1 that is incremented for each parent specifier provided. The driver shim looks for parent-node-*n* attributes. When found, the attribute is checked to see if the named node exists. If the node is found, it uses as the parent for the `<operation-data>` element on the response.

5.5.2 Using Operation Data to Override Default Subscriber Options

There are two ways to override default Subscriber options for a command:

- "Creating and Using Multiple Subscriber Option Sets (connections)" on page 35
- "Overriding Single Subscriber Options" on page 36

Creating and Using Multiple Subscriber Option Sets (connections)

You can override the default Subscriber option by creating multiple sets of the Subscriber options (called connections) in your configuration, and using the `<operation-data>` element to specify which connection set to use for the current command.

To use the `<operation-data>` element to override the default Subscriber connection parameters:

1. Edit the *Subscriber Settings* section of the driver configuration.
2. Using the XML edit feature of iManager, find each Subscriber setting that ends with a dash and the number 1, such as subURL-1, duplicate it, and increment the number.

   For example: `subURL-2`
3. Edit the values of the new settings to be the values you want to use for the second connection.

   You can configure any number of connections this way as long as the numbers you use are incremental without gaps.
4. Add an attribute to the `<operation-data>` element called `connection` and give it the value of the connection number you want to user.

   For example:

```
<operation-data connection="2">
...(other operation-data elements)
</operation-data>
```

Overriding Single Subscriber Options

Instead of using the concept of connections to override multiple Subscriber options, you can override only the url, the HTTP method, or the soap-action values, by directly using attributes on an <operation-data> element. The following table lists the attributes that can be used and the Subscriber option they are meant to override.

Table 5-1 *Attributes Used to Override the Subscriber Options*

<operation-data> Attribute	Subscriber Option Being Overridden	Description
url	subURL-1	This is the URL or (or URI) of the Web Service or HTTP application. Overriding the URL might be useful if, the application has one Web Service for adding a user and a different Web Service for deleting a user.
method	subHttpMethod-1	This is POST by default but can be set to other methods as defined in RFC 2616 Section 9.
soap-action	HTTP Request-Header field with key “SOAPAction”	With the DSML and SPML samples, this value is always #batchRequest. However, there are some Web services that require this value to change, depending on the command.

Examples:

```
<operation-data url="http://137.66.10.13:18180/soap">
...(other operation-data elements if required)
</operation-data>

<operation-data method="GET">
...(other operation-data elements if required)
</operation-data>

<operation-data soap-action="addUser">
...(other operation-data elements if required)
</operation-data>
```

Activating the Driver

6

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

To activate the driver, see "Activating Novell Identity Manager Products" in the *Identity Manager 3.5.1 Installation Guide*.

Managing the Driver

7

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

- Section 7.1, “Starting, Stopping, or Restarting the Driver,” on page 39
- Section 7.2, “Migrating and Resynchronizing Data,” on page 40
- Section 7.3, “Using the DirXML Command Line Utility,” on page 40
- Section 7.4, “Viewing Driver Versioning Information,” on page 41
- Section 7.5, “Reassociating a Driver Set Object with a Server Object,” on page 46
- Section 7.6, “Changing the Driver Configuration,” on page 46
- Section 7.7, “Storing Driver Passwords Securely with Named Passwords,” on page 46
- Section 7.8, “Adding a Driver Heartbeat,” on page 53

7.1 Starting, Stopping, or Restarting the Driver

- Section 7.1.1, “Starting the Driver in Designer,” on page 39
- Section 7.1.2, “Starting the Driver in iManager,” on page 39
- Section 7.1.3, “Stopping the Driver in Designer,” on page 39
- Section 7.1.4, “Stopping the Driver in iManager,” on page 39
- Section 7.1.5, “Restarting the Driver in Designer,” on page 40
- Section 7.1.6, “Restarting the Driver in iManager,” on page 40

7.1.1 Starting the Driver in Designer

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

7.1.2 Starting the Driver in iManager

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

7.1.3 Stopping the Driver in Designer

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

7.1.4 Stopping the Driver in iManager

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

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

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

7.1.5 Restarting the Driver in Designer

1 Open a project in the Modeler, then right-click the driver line.

2 Select *Live* > *Restart Driver*.

7.1.6 Restarting the Driver in iManager

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

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

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

7.2 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:

- **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 (usually a Web Service) can be queried for entries that match the criteria in the publisher filter. However, because of the general nature of the SOAP driver the method for querying the Web Service (if there is one) is not known to the driver shim. Therefore, this feature does not usually work with the SOAP driver.
- **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 explained above:

1 In iManager, 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.

4 Click the appropriate migration button.

For more information, see Chapter 8, “Synchronizing Objects,” on page 55.

7.3 Using the DirXML Command Line Utility

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

For example, you could create a shell script on Linux to check the status of the driver. See Appendix B, "DirXML Command Line Utility," on page 83 for detailed information about the DirXML Command Line utility. For daily tasks, use iManager or Designer.

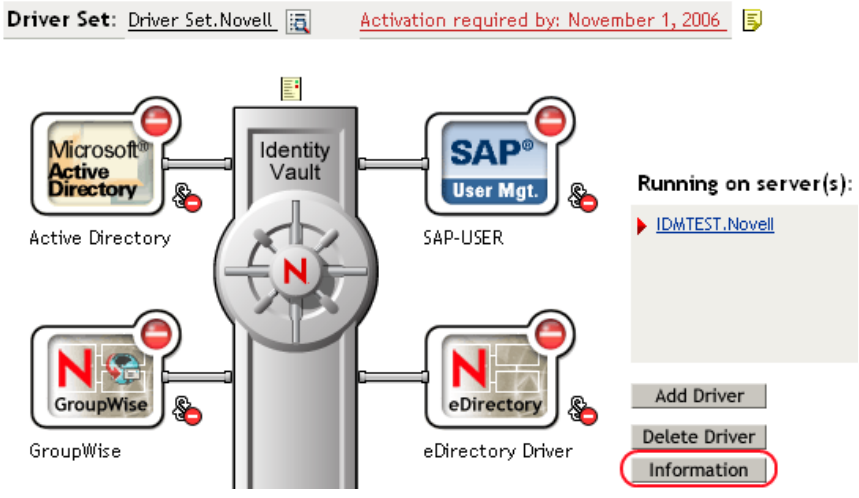
7.4 Viewing Driver Versioning Information

The Versioning Discovery tool only exists in iManager.

- Section 7.4.1, "Viewing a Hierarchical Display of Versioning Information," on page 41
- Section 7.4.2, "Viewing the Versioning Information as a Text File," on page 43
- Section 7.4.3, "Saving Versioning Information," on page 45

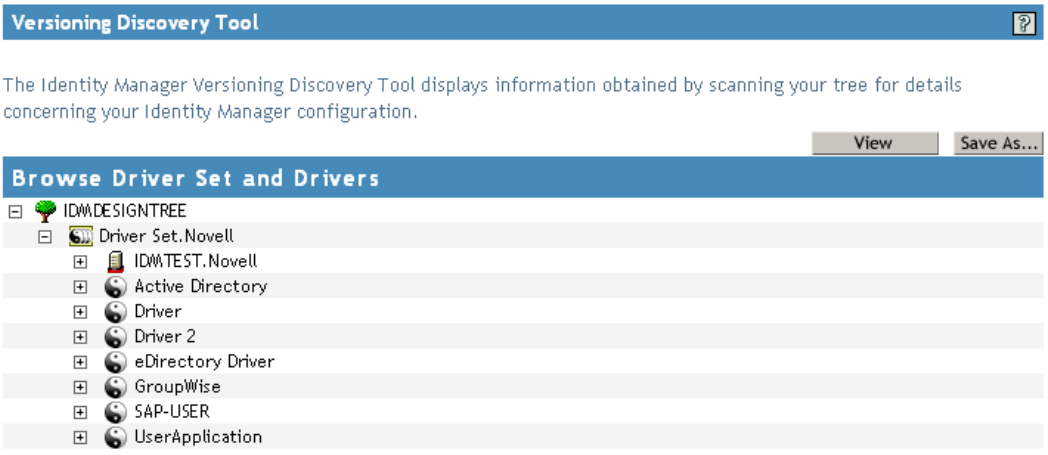
7.4.1 Viewing a Hierarchical Display of Versioning Information

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



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

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

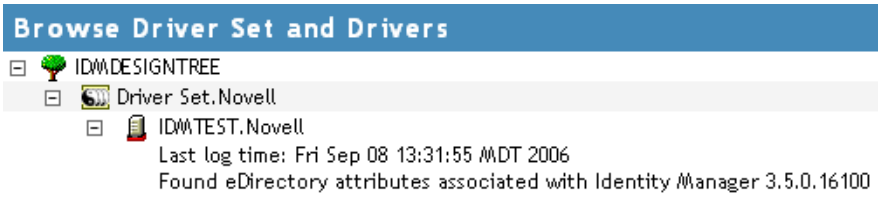


The unexpanded hierarchical view displays the following:

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

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

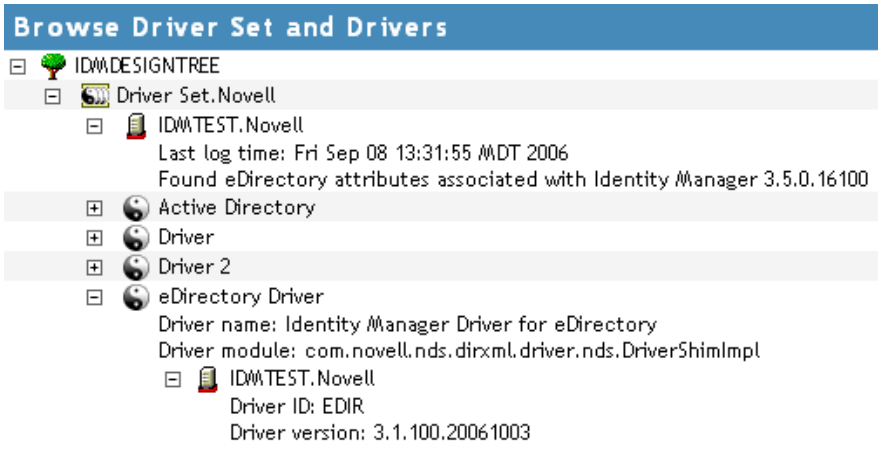
4 View versioning information related to servers by expanding the server icon.



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

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

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



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

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

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

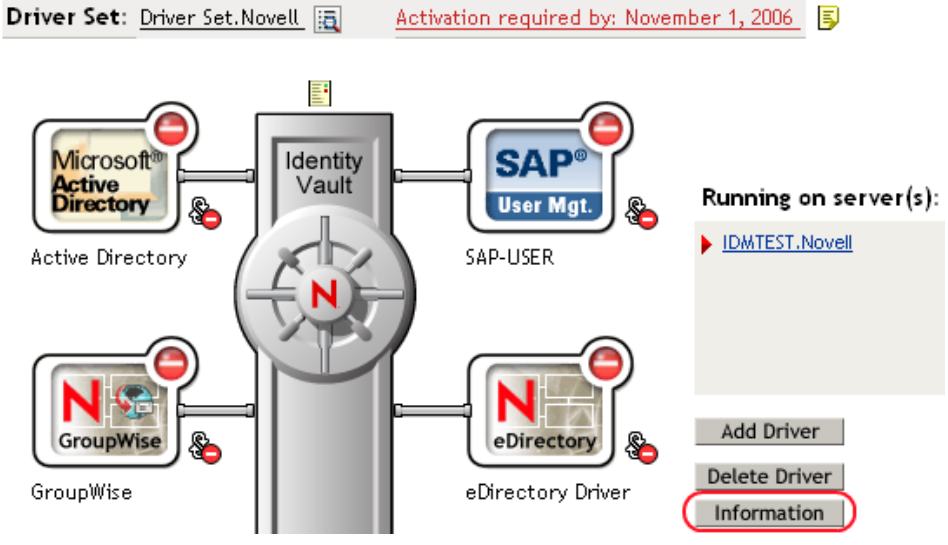
- The driver ID
- The version of the instance of the driver running on that server

7.4.2 Viewing the Versioning Information as a Text File

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

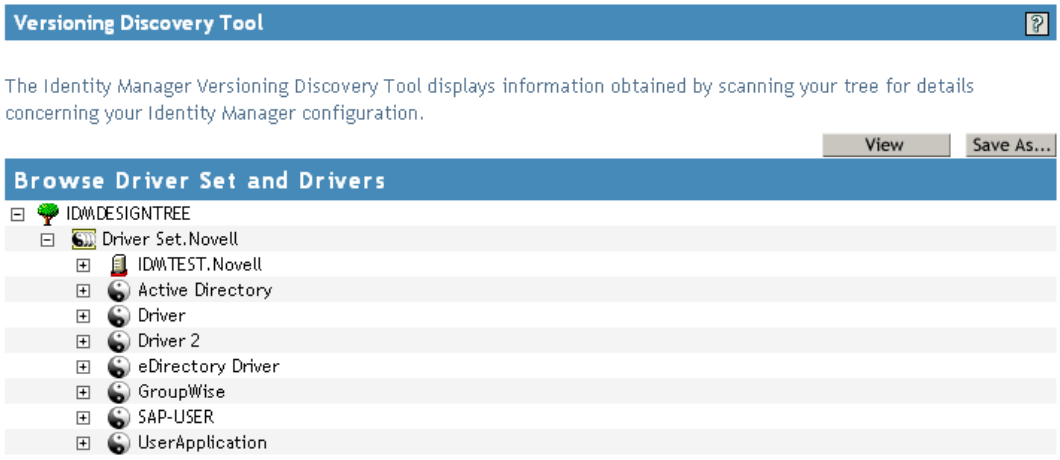
1 To find your Driver Set object in iManager, click *Identity Manager* > *Identity Manager Overview*, then click *Search*.

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

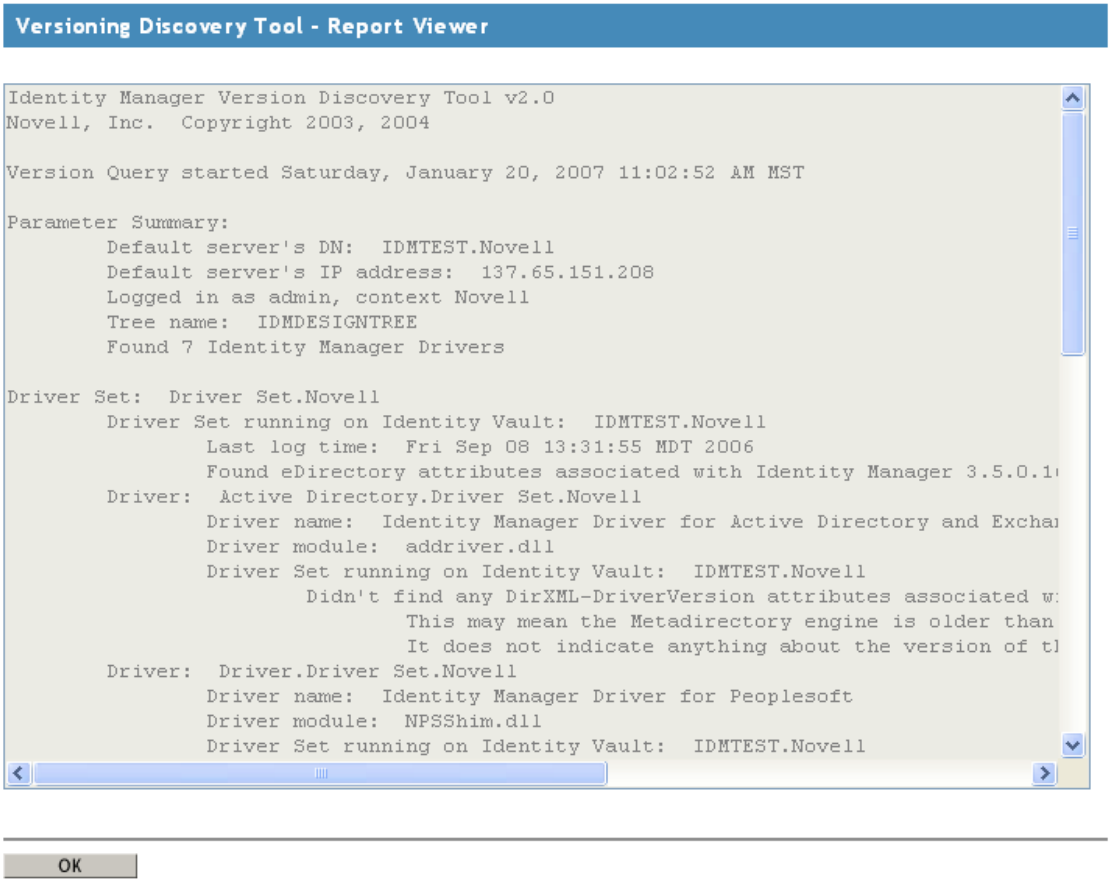


You can also select *Identity Manager Utilities* > *Versioning Discovery*, browse to and select the Driver Set object, then click *Information*.

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



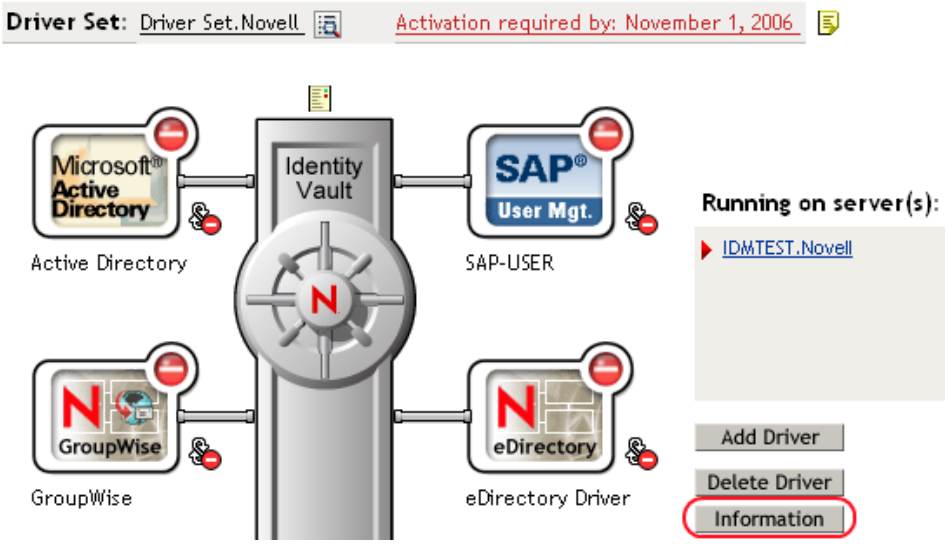
The information is displayed as a text file in the Report Viewer window.



7.4.3 Saving Versioning Information

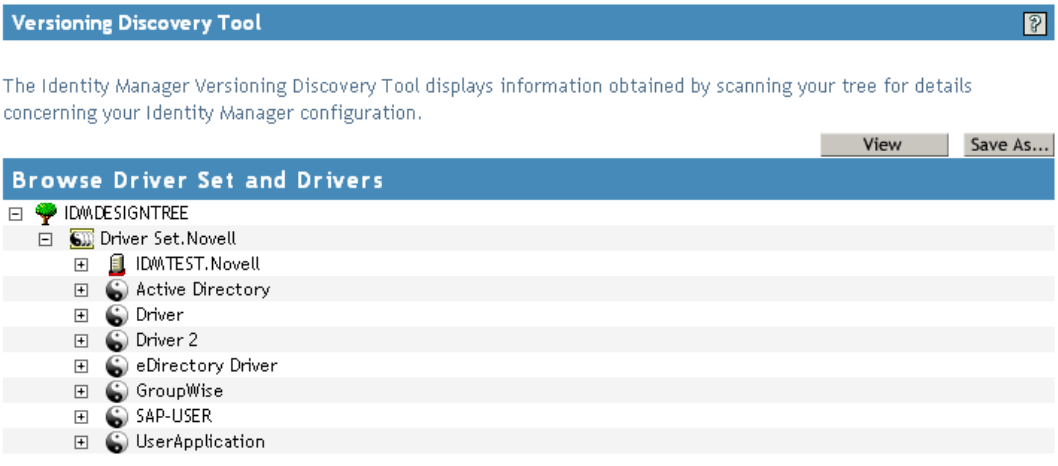
You can save versioning information to a text file on your local or network drive.

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



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

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



4. In the File Download dialog box, click *Save*.
5. Navigate to the desired directory, type a filename, then click *Save*.

   Identity Manager saves the data to a text file.

7.5 Reassociating a Driver Set Object with a Server Object

The driver set object should always be associated with a server object. If the driver set is not associated with a server object, none of the drivers in the driver set can start.

If the link between the driver set object and the server object becomes invalid, you see one of the following conditions:

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

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

1. In iManager click *Identity Manager* > *Identity Manager Overview*, then click *Search* to find the driver set object that the driver should be associated with.
2. Click the *Remove server* icon, then click *OK*.
3. Click the *Add server* icon, then browse to and select the server object.
4. Click *OK*.

7.6 Changing the Driver Configuration

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

To change the driver configuration in iManager:

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

To change the driver configuration in Designer:

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

For a listing of all of the configuration fields, see Appendix C, “Properties of the Driver,” on page 97.

7.7 Storing Driver Passwords Securely with Named Passwords

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

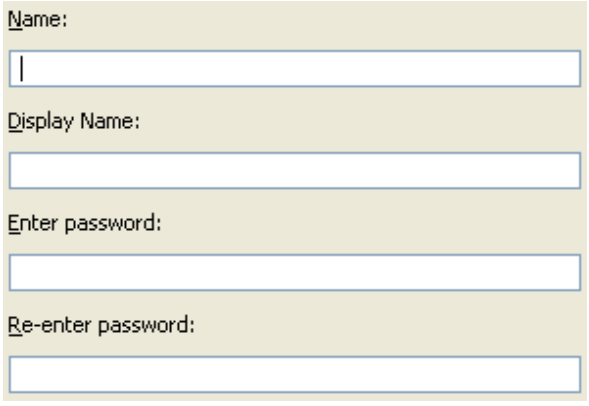
You can also use the Named Passwords feature to store other pieces of information securely, such as a user name.

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

- Section 7.7.1, “Using Designer to Configure Named Passwords,” on page 47
- Section 7.7.2, “Using iManager to Configure Named Passwords,” on page 47
- Section 7.7.3, “Using Named Passwords in Driver Policies,” on page 49
- Section 7.7.4, “Using the DirXML Command Line Utility to Configure Named Passwords,” on page 49

7.7.1 Using Designer to Configure Named Passwords

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



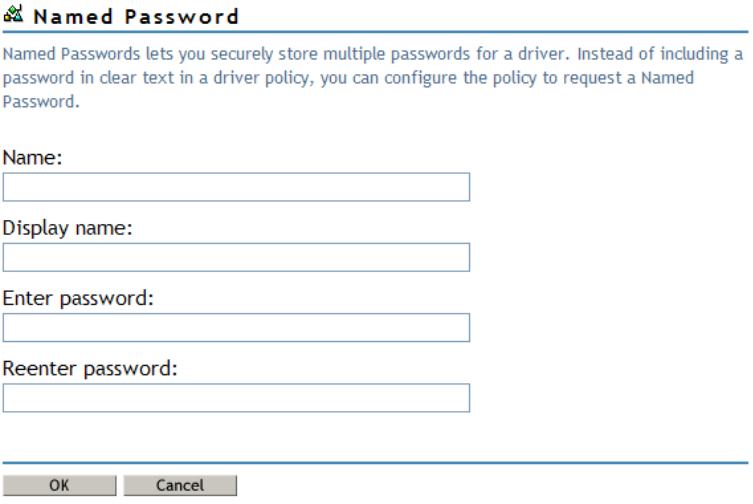
3. Specify the *Name* of the Named Password.
4. Specify the *Display name* of the Named Password.
5. Specify the Named Password, then re-enter the password.
6. Click *OK* twice.

7.7.2 Using iManager to Configure Named Passwords

1. Click *Identity Manager* > *Identity Manager Overview*, then click *Search* to search for the driver set that is associated with the driver.
2. In the Identity Manager Overview, click the upper right corner of the driver icon, then click *Edit properties*.
3. On the Modify Object page on the Identity Manager tab, click *Named Passwords*.

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

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



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

You can use this feature to store other kinds of information securely, such as a username.

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

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

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

7.7.3 Using Named Passwords in Driver Policies

- "Using the Policy Builder" on page 49
- "Using XSLT" on page 49

Using the Policy Builder

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

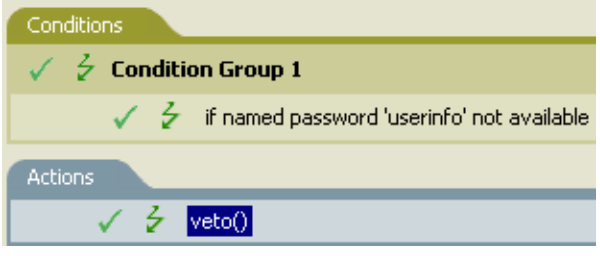
1. In Designer, launch Policy Builder, right-click, then click *New* > *Rule*.
2. Specify the name of the rule, then click *Next*.
3. Select the condition structure, then click *Next*.
4. Select *named password* for the *Condition*.
5. Browse to and select the Named Password that is stored on the driver.

   In this example, it is *userinfo*.
6. Select whether the Operator is available or not available.
7. Select an action for the *Do* field.

   In this example, the action is *veto*.

The example indicates that if the userinfo Named Password is not available, then the event is vetoed.

Figure 7-1 *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/java/
com.novell.nds.dirxml.driver.XdsQueryProcessor/>
```

7.7.4 Using the DirXML Command Line Utility to Configure Named Passwords

- "Creating a Named Password in the DirXML Command Line Utility" on page 50
- "Using the DirXML Command Line Utility to Remove a Named Password" on page 51

Creating a Named Password in the DirXML Command Line Utility

1. Run the DirXML Command Line utility.

   For information, see Appendix B, “DirXML Command Line Utility,” on page 83.

2. Enter your username and password.

   The following list of options appears.

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

3. Enter `3` for driver operations.

   A numbered list of drivers appears.

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

   The following list of options appears.

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

5. Enter `13` for password operations.

   The following list of options appears.

```
Select a password operation
 1: Set shim password
 2: Reset shim password
 3: Set Remote Loader password
 4: Clear Remote Loader password
 5: Set named password
 6: Clear named password(s)
 7: List named passwords
```

```
 8: Get passwords state
99: Exit
Enter choice:
```

6 Enter `5` to set a new Named Password.

The following prompt appears:

```
Enter password name:
```

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

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

```
Enter password:
```

The characters you type for the password are not displayed.

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

```
Confirm password:
```

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

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

Using the DirXML Command Line Utility to Remove a Named Password

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

1 Run the DirXML Command Line utility.

For information, see Appendix B, “DirXML Command Line Utility,” on page 83.

2 Enter your username and password.

The following list of options appears.

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

3 Enter `3` for driver operations.

A numbered list of drivers appears.

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

The following list of options appears.

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

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

5 Enter `13` for password operations.

The following list of options appears.

```
Select a password operation
 1: Set shim password
 2: Reset shim password
 3: Set Remote Loader password
 4: Clear Remote Loader passwor
 5: Set named password
 6: Clear named password(s)
 7: List named passwords
 8: Get passwords state
99: Exit
Enter choice:
```

6 (Optional) Enter `7` to see the list of existing Named Passwords.

The list of existing Named Passwords is displayed.

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

7 Enter `6` to remove one or more Named Passwords.

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

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

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

```
Enter password name:
```

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

```
Select a password operation
 1: Set shim password
 2: Reset shim password
 3: Set Remote Loader password
 4: Clear Remote Loader password
 5: Set named password
 6: Clear named password(s)
 7: List named passwords
 8: Get passwords state
99: Exit
Enter choice:
```

10 (Optional) Enter `7` to see the list of existing Named Passwords.

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

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

7.8 Adding a Driver Heartbeat

The driver heartbeat is a feature of the Identity Manager drivers that ship with Identity Manager 2 and later. Its use is optional. The driver heartbeat is configured by using a driver parameter with a time interval specified. If a heartbeat parameter exists and has an interval value other than 0, the driver sends a heartbeat document to the Metadirectory engine if there is no communication on the Publisher channel for the specified interval of time.

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

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

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

To configure the heartbeat:

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

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

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

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

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

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

After you have added the driver parameter, you can edit the time interval by using the graphical view. Another option is to create a reference to a global configuration value (GCV) for the time interval. Like other global configuration values, the driver heartbeat can be set at the driver set level

instead of on each individual driver object. If a driver does not have a particular global configuration value, and the driver set object does have it, the driver inherits the value from the driver set object.

Synchronizing Objects

8

This section explains driver and object synchronization in DirXML® 1.1a, Identity Manager 2.0, and Identity Manager 3.x. Driver synchronization was not available for DirXML 1.0 and DirXML 1.1.

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

- Section 8.1, “What Is Synchronization?,” on page 55
- Section 8.2, “When Is Synchronization Done?,” on page 55
- Section 8.3, “How Does the Metadirectory Engine Decide Which Object to Synchronize?,” on page 56
- Section 8.4, “How Does Synchronization Work?,” on page 57

8.1 What Is Synchronization?

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

- Synchronization (or merging) of attribute values of an object in the Identity Vault with the corresponding attribute values of an associated object in a connected system.
- Migration of 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.

8.2 When Is Synchronization Done?

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

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

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

- The user issues a manual driver synchronization request. This corresponds to the *Resync* button in the Driver Set property page in ConsoleOne®, or to the *Synchronize* button on the iManager Identity Manager Driver Overview page.
- The Metadirectory engine encounters an error with the driver's cache and cannot recover from the cache error. The driver's cache is deleted and the engine generates object synchronization commands as detailed in Section 8.3, "How Does the Metadirectory Engine Decide Which Object to Synchronize?," on page 56.

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

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

The starting filter time is used to filter objects that have modification or creation times that are older than the starting time specified 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 haven't yet been successfully processed by the driver's Subscriber channel.

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

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

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

     and
   - Exist in the filter on the Subscriber channel.

2. It finds all objects that have an entry creation time stamp greater than or equal to the starting filter time.
3. It adds a `synchronize object` command to the driver cache for each unique object found that has an entry modification time stamp greater than or equal to the starting filter time and all objects and classes that are in the Subscriber filter channel in the driver being synchronized.

8.4 How Does Synchronization Work?

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

1. Each system (the Identity Vault and the connected system) is queried for all attribute values in the appropriate filters.
   - eDirectory is queried for all values in the Subscriber filter, and for values that are marked for synchronization in Identity Manager 2.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 8-1 on page 58, Table 8-2 on page 59, and Table 8-3 on page 61.

   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.

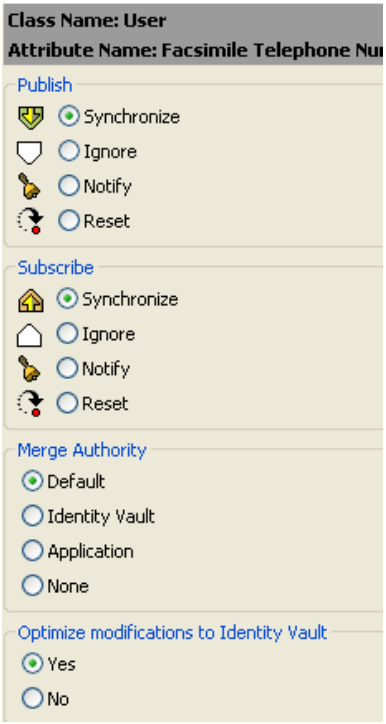
There are three different combinations of selected items in the filter, and each one creates a different output.

- Section 8.4.1, “Scenario One,” on page 57
- Section 8.4.2, “Scenario Two,” on page 59
- Section 8.4.3, “Scenario Three,” on page 60

8.4.1 Scenario One

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

Figure 8-1 *Scenario One*



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

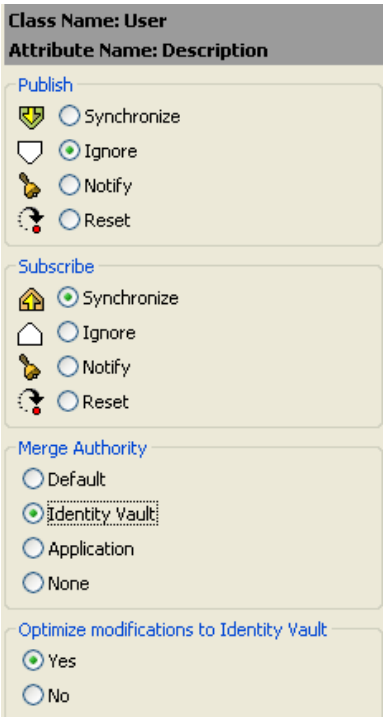
Table 8-1 *Output of Scenario One*

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

8.4.2 Scenario Two

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

Figure 8-2 *Scenario Two*



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

Table 8-2 *Output of Scenario Two*

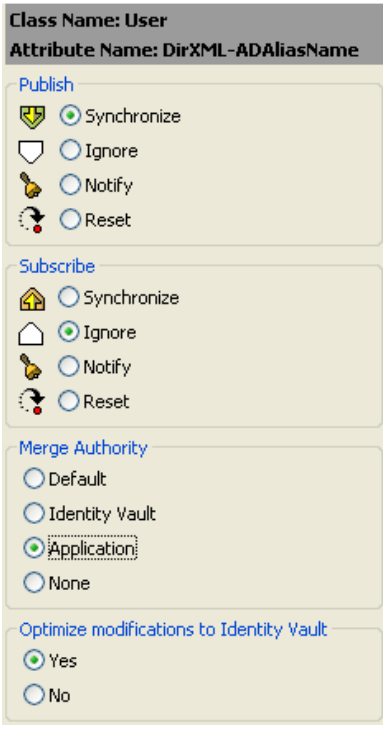
	Identity Vault single-valued empty	**Identity Vault single-valued non-empty**	**Identity Vault multi-valued empty**	**Identity Vault multi-valued non-empty**
Application single-valued empty	No change	App = Identity Vault	No change	App = Identity Vault[1]
Application single-valued empty	App = empty	App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

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

8.4.3 Scenario Three

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

Figure 8-3 *Scenario Three*



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

Table 8-3 *Output of Scenario Three*

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

Troubleshooting the Driver

9

You can log Identity Manager events using Novell® Audit. 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 "Integrating Identity Manager with Novell Audit" in the *Identity Manager 3.5.1 Logging and Reporting*.

This section contains the following information on error messages:

- Section 9.1, "Driver Shim Errors," on page 63
- Section 9.2, "Java Customization Errors," on page 66
- Section 9.3, "Troubleshooting Driver Processes," on page 67

9.1 Driver Shim Errors

The following 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. See Section 5.2, "Configuring Subscriber Settings," on page 32 for more information.

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 doesn't 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 Metadirectory 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. See Chapter 5, "Configuring the Driver," on page 31 for more information.

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. See Section 5.3, "Configuring Publisher Settings," on page 33 for more information.

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. See Section 5.3, "Configuring Publisher Settings," on page 33 for more information.

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. See Section 5.3, “Configuring Publisher Settings,” on page 33 for more information.

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

9.2 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 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 table 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

9.3 Troubleshooting Driver Processes

Viewing driver processes is necessary to analyze unexpected behavior. To view the driver processing events, use DSTrace. You should only use it during testing and troubleshooting the driver. Running DSTrace while the drivers are in production increases the utilization on the Identity Manager server and can cause events to process very slowly.

9.3.1 Viewing Driver Processes

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

- “Adding Trace Levels in Designer” on page 67
- “Adding Trace Levels in iManager” on page 69
- “Capturing Driver Processes to a File” on page 70

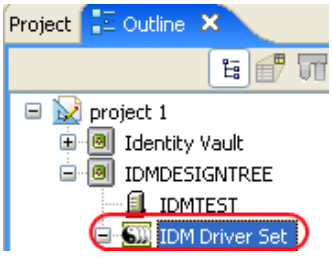
Adding Trace Levels in Designer

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

- “Driver Set” on page 67
- “Driver” on page 68

Driver Set

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



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

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

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

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

Driver

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

Parameter	Description
Trace level	As the driver object trace level increases, the amount of information displayed in DSTrace increases. Trace level 1 shows errors, but not the cause of the errors. If you want to see password synchronization information, set the trace level to 5. If you select *Use setting from Driver Set*, the value is taken from the driver set object.
Trace file	Specify a filename and location for where the Identity Manager information is written for the selected driver. If you select *Use setting from Driver Set*, the value is taken from the driver set object.

Parameter	Description
Trace file size limit	Allows you to set a limit for the Java trace file. If you set the file size to *Unlimited*, the file grows in size until there is no disk space left. If you select *Use setting from Driver Set*, the value is taken from the driver set object. **NOTE:** The trace file is created in multiple files.Identity Manager automatically divides the maximum file size by ten and creates ten separate files. The combined size of these files equals the maximum trace file size.
Trace name	The driver trace messages are prepended with the value entered instead of the driver name. Use this option if the driver name is very long.

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

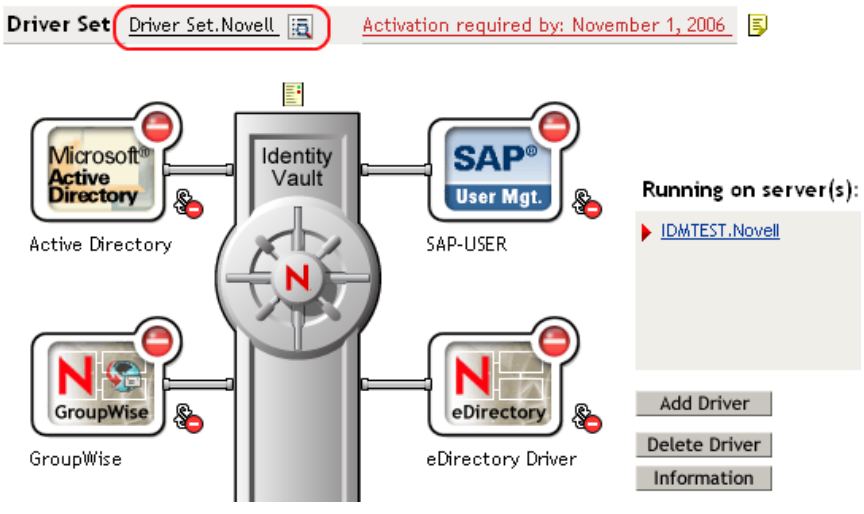
Adding Trace Levels in iManager

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

- “Driver Set” on page 69
- “Driver” on page 70

Driver Set

1. In iManager, select *Identity Manager* > *Identity Manager Overview*.
2. Browse to the driver set object, then click *Search*.
3. Click the driver set name.



4. Select the *Misc* tab for the driver set object.
5. Set the parameters for tracing, then click *OK*.

   See “Misc” on page 111 for the parameters.

Driver

1. In iManager, select *Identity Manager* > *Identity Manager Overview*.
2. Browse to the driver set object where the driver object resides, then click *Search*.
3. Click the upper right corner of the driver object, then click *Edit properties*.
4. Select the *Misc* tab for the driver object.
5. Set the parameters for tracing, then click *OK*.

   See "Misc" on page 111 for the parameters.

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

Capturing Driver Processes to a File

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

The 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 helps you capture and save Identity Manager processes through DSTrace on different platforms.

- "NetWare" on page 70
- "Windows" on page 71
- "UNIX" on page 71
- "iMonitor" on page 71
- "Remote Loader" on page 72

NetWare

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

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

This stops capturing trace messages to the log file. It also stops logging information into the file.

10 Open the `dstrace.log` in a text editor and search for the event or the object you modified.

Windows

1 Open the *Control Panel* > *NDS Services* > `dstrace.dlm`, then click *Start* to display the NDS Server Trace utility window.

2 Click *Edit* > *Options*, then click *Clear All* to clear all of the default flags.

3 Select *DirXML* and *DirXML Drivers*.

4 Click *OK*.

5 Click *File* > *New*.

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

7 Wait for the event to occur.

8 Click *File* > *Close*.

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

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

UNIX

1 Enter `ndstrace` to start the ndstrace utility.

2 Enter `set ndstrace=nodebug` to turn off all trace flags currently set.

3 Enter `set ndstrace on` to display trace messages to the console.

4 Enter `set ndstrace file on` to capture trace messages to the `ndstrace.log` file in the directory where eDirectory is installed. By default it is `/var/nds`.

5 Enter `set ndstrace=+dxml` to display the Identity Manager events.

6 Enter `set ndstrace=+dvrs` to display the Identity Manager driver events.

7 Wait for the event to occur.

8 Enter `set ndstrace file off` to stop logging information to the file.

9 Enter `exit` to quite the ndstrace utility.

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

iMonitor

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

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

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

Port 8008 is the default.

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

3 Select *Trace Configuration* on the left side.

4 Click *Clear All*.

5 Select *DirXML* and *DirXML Drivers*.

6 Click *Trace On*.

7 Select *Trace History* on the left side.

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

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

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

11 Select *Trace History* to view the trace history.

The files are distinguished by their time stamp.

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

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

Remote Loader

You can capture the events that occur on the machine by 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 Table 9-1. For more information, see "Configuring the Remote Loader " in the *Novell Identity Manager 3.5.1 Administration Guide*.

Table 9-1 *Command Line Tracing Switches*

Option	Short Name	Parameter	Description
-trace	-t	integer	Specifies the trace level. This is only used when hosting an application shim. Trace levels correspond to those used on the Identity Manager server. Example: `-trace 3` or `-t3`
-tracefile	-tf	filename	Specifies a file to write trace messages to. Trace messages are written to the file if the trace level is greater than zero. Trace messages are written to the file even if the trace window is not open. Example: `-tracefile c:\temp\trace.txt` or `-tf c:\temp\trace.txt`

Option	Short Name	Parameter	Description
-tracefilemax	-tfm	size	Specifies the approximate maximum size that trace file data can occupy on disk. If you specify this option, there is a trace file with the name specified using the tracefile option and up to 9 additional “roll-over” files. The roll-over files are named using the base of the main trace filename plus “_n”, where n is 1 through 9. The size parameter is the number of bytes. Specify the size by using the suffixes K, M, or G for kilobytes, megabytes, or gigabytes. If the trace file data is larger than the specified maximum when the Remote Loader is started, the trace file data remains larger than the specified maximum until roll-over is completed through all 10 files. Example: `-tracefilemax 1000M` or `-tfm 1000M`

Backing Up the Driver

10

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 *Designer 2.1 for Identity Manager 3.5.1*.

- Section 10.1, "Exporting the Driver in Designer," on page 75
- Section 10.2, "Exporting the Driver in iManager," on page 75

10.1 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.

10.2 Exporting the Driver in iManager

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

Security: Best Practices

11

For more information on how to secure the driver and the information it is synchronizing, see "Security: Best Practices" in the *Novell Identity Manager 3.5.1 Administration Guide*.

Using Java Extensions

A

The functionality of the Identity Manager Driver for SOAP can be extended by using Java. Using an API defined by Java interfaces, you can create your own custom Java classes that have access to the data passing through the Subscriber channel and Publisher channel. These classes can read and interpret the data, and, optionally, can modify the data. There are also Java interfaces defined to let you replace the default subscriber or publisher (that uses HTTP) with your own custom subscriber or publisher.

This section contains the following information on using Java extensions:

- Section A.1, “Overview,” on page 79
- Section A.2, “Creating and Configuring Java Extensions,” on page 80

A.1 Overview

If the application you are using with the Identity Manager Driver for SOAP uses non-XML data, you can create Java extensions to convert the non-XML data to XML data. Or, you might want to change various protocols, including XML and HTTP. For example, the default HTTP can be replaced. These Java extensions can be used to operate on data and they must be used to convert non-XML data to XML data. As illustrated in Figure A-1 on page 80, there are eleven points where functionality can be extended:

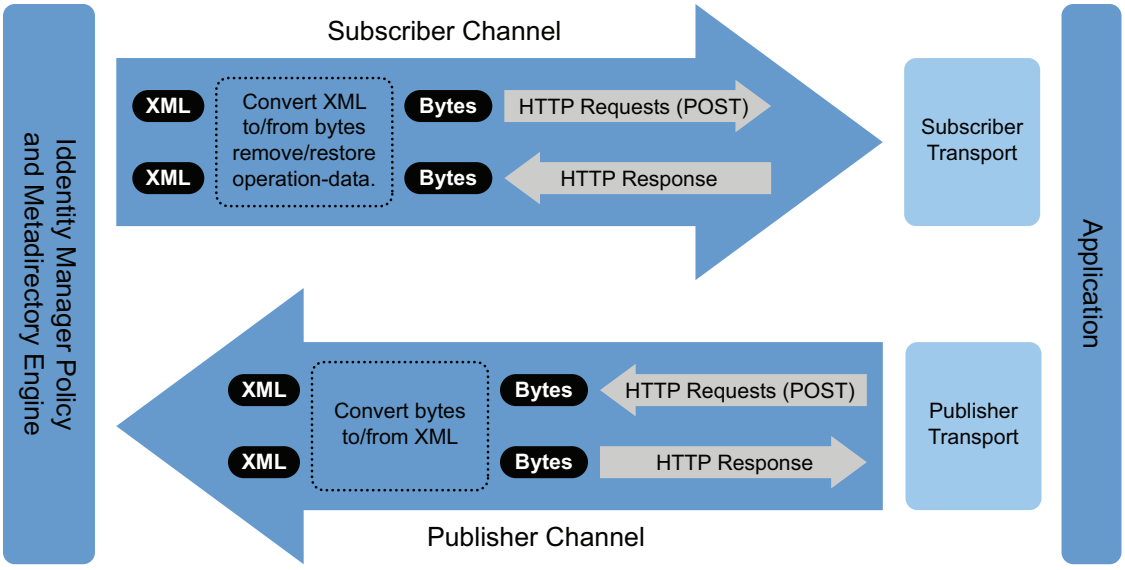
- Four in the Subscriber channel
- Four in the Publisher channel
- Two to specify the transport
- One to report the application schema

The SOAP driver was designed to be flexible and extensible. For the Java programmer who wants to extend or modify the capabilities of the driver, there are programming interfaces that can be used for this purpose. These interfaces should be used only when you need to do transformations that cannot be done in policies or style sheets.

The Javadoc (http://www.novell.com/documentation/beta/dirxmldrivers/javadoc/api/index.html) describes these interfaces.

There are five Java interfaces that can be used to extend or customize the driver behavior. They are DocumentModifiers, ByteArrayModifiers, PublisherTransport, SubscriberTransport, and SchemaReporter.

Figure A-1 *How Functionality Can Be Extended Using Java*



DocumentModifiers and ByteArrayModifiers serve a similar purpose, so you should probably use one or the other. They are both used to access and to modify the commands and events passing through the driver shim, if this is desired. DocumentModifiers gives you access to the data as XML DOM documents. ByteArrayModifiers gives you access to the same data, but serialized as byte arrays.

The PublisherTransport interface allows you to replace the default HTTP listener that the driver uses on the Publisher channel with something else. Your PublisherTransport implementation can either be event driven, or it can poll at a specified interval.

If you want to replace the HTTP or HTTPS connections that the driver uses on the Subscriber channel with something else, you would implement a SubscriberTransport.

The remaining interface, SchemaReporter, can be used if you have a way of programatically determining the classes and attributes used by the remote Web service. The advantage to this is that creating schema mapping rules is easier if the schema can be dynamically determined.

A.2 Creating and Configuring Java Extensions

Using the sample code and SOAP Driver Javadoc found at the Novell Developer Downloads Web site (http://developer.novell.com/ndk/downloadaz.htm) as a guide, write the Java code for your class. In the A-Z listing, search for SOAP Driver. You should name your class using any Java package and class name that is convenient to your environment and your organization.

For example, if you were writing your own class that implemented the DocumentModifiers interface, and you named your class *MyDocumentModifiers* within a package called com.novell.idm, then you would perform the following steps to compile, jar, and deploy your class:

1. Prepare your environment.

   Make sure you have a current Java Development Kit (JDK) installed on your computer. Visit the Java Web Site (http://java.sun.com/) if you need to download one.

2. Gather your source code in the proper directory structure as defined by your package naming.

In the example given above, you would have a `com` directory that contained a `novell` directory that contained an `idm` directory. Within the `idm` directory, you would have a source file named `MyDocumentModifiers.java`.

3. Make sure you have the jar files you need to compile your class.

   At a minimum, you need `SOAPUtil.jar`. If you are using XML documents within your class, you also need `nxsl.jar`.

4. Put a copy of the required jar files in a convenient location like the root of your compile directory just outside the `com` directory, then access a system command prompt or shell prompt with that location as the current directory.

5. Compile your class by entering one of the following:

   - **For Windows:** `javac -classpath SOAPUtil.jar;nxsl.jar com\novell\idm\*.java`
   - **For Linux or UNIX:** `javac -classpath SOAPUtil.jar:nxsl.jar com/novell/idm/*.java`

6. Create a Java archive file containing your class by entering one of the following:

   - **For Windows:** `jar cvf mydriverextensions.jar com\novell\idm\*.class`
   - **For Linux:** `jar cvf mydriverextensions.jar com/novell/idm/*.class`

7. Place the jar file you created in Step 6 into the same directory that contains the `SOAPShim.jar`.

   In Windows this is often `C:\Novell\NDS\lib`.

8. In iManager, edit the driver settings.

   8a Next to Custom Java Extension, select *Show*.

   8b Next to Document Handling, select *Implemented*.

   8c Specify *com.novell.idm.MyDocumentModifiers* as the value for Class and any string as the value for Init Parameter.

   The init parameter is the string that is passed to the init method of your class, so put any information here that you want to use during your class initialization.

9. Restart the driver.

You can now use your custom class.

DirXML Command Line Utility

B

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

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

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

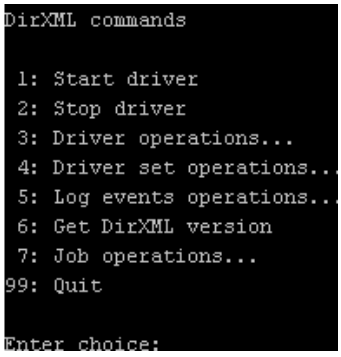
There are two different methods for using the DirXML Command Line utility:

- Section B.1, “Interactive Mode,” on page 83
- Section B.2, “Command Line Mode,” on page 92

B.1 Interactive Mode

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

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



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

   Table B-1 on page 84 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 B-1 *Interactive Mode Options*

Option	Description
1: *Start Driver*	Starts the driver. If there is more than one driver, each driver is listed with a number. Enter the number of the driver to start the driver.
2: *Stop Driver*	Stops the driver. If there is more than one driver, each driver is listed with a number. Enter the number of the driver to stop the driver.
3: *Driver operations*	Lists the operations available for the driver. If there is more than one driver, each driver is listed with a number. Enter the number of the driver to see the operations available. See Table B-2 on page 85 for a list of operations.
4: *Driver set operations*	Lists the operations available for the driver set. ◆ 1: Associate driver set with server ◆ 2: Disassociate driver set from server ◆ 99: Exit
5: *Log events operations*	Lists the operations available for logging events through Novell® Audit. See Table B-5 on page 89 for a description of these options.
6: *Get DirXML version*	Lists the version of the Identity Manager installed.
7: *Job operations*	Manages jobs created for Identity Manager.
99: *Quit*	Exits the DirXML Command Line utility

Figure B-1 *Driver Options*

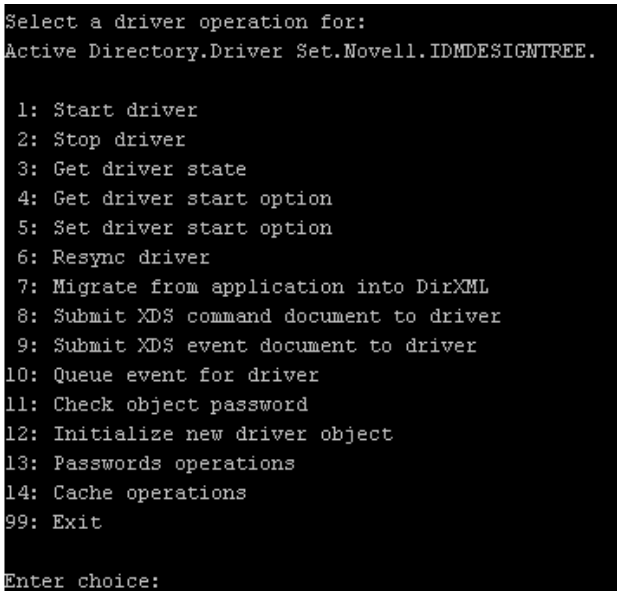


Table B-2 *Driver Options*

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

Options	Description
8: *Submit XDS command document to driver*	Processes an XDS command document: *Enter filename of XDS command document:* Examples: NetWare: `sys:\files\user.xml` Windows: `c:\files\user.xml` Linux: `/files/user.xml` *Enter name of file for response:* Examples: NetWare: `sys:\files\user.log` Windows: `c:\files\user.log` Linux: `/files/user.log`
9: Submit XDS event document to driver	Processes an XDS event document: *Enter filename of XDS event document:* Examples: NetWare: `sys:\files\add.xml` Windows: `c:\files\add.xml` Linux: `/files/add.xml`
10: Queue event for driver	Adds an event to the driver queue: *Enter filename of XDS event document:* Examples: NetWare: `sys:\files\add.xml` Windows: `c:\files\add.xml` Linux: `/files/add.xml`
11: *Check object password*	Validates that an object's password in the connected system is associated with a driver. It matches the object's eDirectory password (Distribution Password, used with Universal Password). *Enter user name*:
12: *Initialize new driver object*	Performs an internal initialization of data on a new Driver object. This is only for testing purposes.
13: *Password operations*	There are nine Password options. See Table B-3 on page 87 for a description of these options.
14: *Cache operations*	There are five Cache operations. See Table B-4 on page 88 for a descriptions of these options.

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

Figure B-2 *Password Operations*

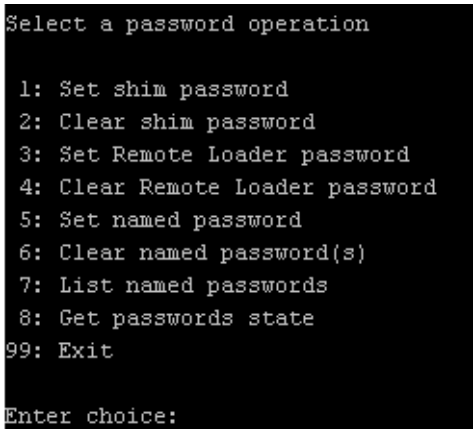


Table B-3 *Password Operations*

Operation	Description
1: *Set shim password*	Sets the application password. This is the password of the user account you are using to authenticate into the connected system with.
2: *Clear shim password*	Clears the application password.
3: *Set Remote Loader password*	The Remote Loader password is used to control access to the Remote Loader instance. Enter the Remote Loader password, then confirm the password by typing it again.
4: *Clear Remote Loader password*	Clears the Remote Loader password so no Remote Loader password is set on the Driver object.
5: *Set named password*	Allows you to store a password or other pieces of security information on the driver. See Section 7.7, “Storing Driver Passwords Securely with Named Passwords,” on page 46 for more information. There are four prompts to fill in: ◆ *Enter password name:* ◆ *Enter password description:* ◆ *Enter password:* ◆ *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).* If you enter Yes, all Named Passwords are cleared. If you enter No, you are prompted to specify the password name that you want to clear.
7: *List named passwords*	Lists all Named Passwords that are stored on the driver object. It lists the password name and the password description.
8: *Get password state*	Lists if a password is set for: ◆ Driver Object password ◆ Application password ◆ Remote loader password The dxcmd utility allows you to set the Application password and the Remote Loader password. You cannot set the Driver Object password with this utility. It shows if the password has been set or not.
99: *Exit*	Exits the current menu and takes you back to the Driver options.

Figure B-3 *Cache Operations*

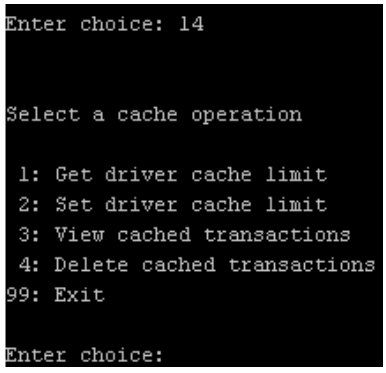


Table B-4 *Cache Operations*

Operation	Description
1: *Get driver cache limit*	Displays the current cache limit that is set for the driver.
2: *Set driver cache limit*	Sets the driver cache limit in kilobytes. A value of 0 is unlimited.

Operation	Description
3: *View cached transactions*	A text file is created with the events that are stored in cache. You can select the number of transactions to view. ◆ *Enter option token* (default=0): ◆ *Enter maximum transactions records to return* (default=1): ◆ *Enter name of file for response:*
4: *Delete cached transactions*	Deletes the transactions stored in cache. ◆ *Enter position token* (default=0): ◆ *Enter event-id value of first transaction record* to delete (optional): ◆ *Enter number of transaction records to delete* (default=1):
99: *Exit*	Exits the current menu and takes you back to the Driver options.

Figure B-4 *Log Event Operations*

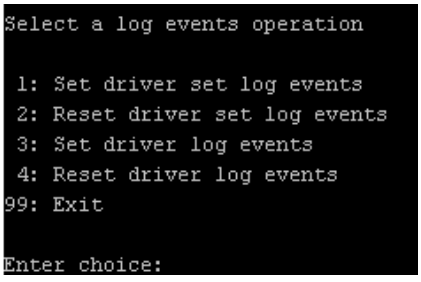


Table B-5 *Log Events Operations*

Operation	Description
1: *Set driver set log events*	Allows you to log driver set events through Novell Audit. There are 49 items you can select to log. See Table B-6 on page 90 for a list of these options. Type the number of the item you want to log. After the items are selected, enter 99 to accept the selections.
2: *Reset driver set log events*	Resets all of the log event options.
3: *Set driver log events*	Allows you to log driver events through Novell Audit. There are 49 items to select to log. See Table B-6 on page 90 for a list of these options. Type the number of the item you want to log. After the items are selected, enter 99 to accept the selections.

Operation	Description
4: *Reset driver log events*	Resets all of the log event options.
99: *Exit*	Exits the log events operations menu.

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

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

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

Table B-7 *Enter Table Title Here*

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

B.2 Command Line Mode

The command line mode allows you to use script or batch files. Table B-8 on page 92 lists the different options that are available.

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

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

This example command starts the driver.

Table B-8 *Command Line Options*

Option	Description
Configuration	
-user <*user name*>	Specify the name of a user with administrative rights to the drivers you want to test.
-host <*name or IP address*>	Specify the IP address of the server where the driver is installed.
-password <*user password*>	Specify the password of the user specified above.
-port <*port number*>	Specify a port number, if the default port is not used.
-q <*quiet mode*>	Displays very little information when a command is executed.
-v <*verbose mode*>	Displays detailed information when a command is executed.

Option	Description
-s <*stdout*>	Writes the results of the `dxcmd` command to `stdout`.
-? <*show this message*>	Displays the help menu.
-help <*show this message*>	Displays the help menu.
Actions	
-start <*driver dn*>	Starts the driver.
-stop <*driver dn*>	Stops the driver.
-getstate <*driver dn*>	Shows the state of the driver as running or stopped.
-getstartoption <*driver dn*>	Shows the startup option of the driver.
-setstartoption <*driver dn*> <*disabled*|*manual*|*auto*> <*resync*|*noresync*>	Sets how the driver starts if the server is rebooted. Sets whether the objects are to be resynchronized when the driver restarts.
-getcachelimit <*driver dn*>	Lists the cache limit set for the driver.
-setcachelimit <*driver dn*> <*0 or positive integer*>	Sets the cache limit for the driver.
-migrateapp <*driver dn*> <*filename*>	Processes an XML document that contains a query command. Create the XML document that contains a query command by using the Novell `nds.dtd` (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_dtd/data/dtdndsoverview.html#dtdndsoverview).
-setshimpassword <*driver dn*> <*password*>	Sets the application password. This is the password of the user account you are using to authenticate into the connected system with.
-clearshimpassword <*driver dn*> <*password*>	Clears the application password.
-setremoteloaderpassword <*driver dn*> <*password*>	Sets the Remote Loader password. The Remote Loader password is used to control access to the Remote Loader instance.
<clearremoteloaderpassword <*driver dn*>	Clears the Remote Loader password.

Option	Description
-sendcommand *<driver dn> <input filename> <output filename>*	Processes an XDS command document. Specify the XDS command document as the input file. Examples: NetWare: `sys:\files\user.xml` Windows: `c:\files\user.xml` Linux: `/files/user.log` Specify the output filename to see the results. Examples: NetWare: `sys:\files\user.log` Windows: `c:\files\user.log` Linux: `/files/user.log`
-sendevent *<driver dn> <input filename>*	Submits a document to the driver's Subscriber channel, bypassing the driver cache. The document is processed ahead of anything that might be in the cache at the time of the submission. It also means that the submission fails if the driver is not running.
-queueevent *<driver dn> <input filename>*	Submits a document to the driver's Subscriber channel by queuing the document in the driver cache. The document gets processed after anything that might be in the cache at the time of the submission. The submission won't fail if the driver isn't running.
-setlogevents *<dn> <integer ...>*	Sets Novell Audit log events on the driver. The integer is the option of the item to log. See Table B-6 on page 90 for the list of the integers to enter.
-clearlogevents *<dn>*	Clears all Novell Audit log events that are set on the driver.
-setdriverset *<driver set dn>*	Associates a driver set with the server.
-cleardriverset	Clears the driver set association from the server.
-getversion	Shows the version of Identity Manager that is installed.
-initdriver object *<dn>*	Performs an internal initialization of data on a new Driver object. This is only for testing purposes.
-setnamedpassword *<driver dn> <name> <password> [description]*	Sets Named Passwords on the driver object. You specify the name, the password, and the description of the Named Password.
-clearnamedpassword *<driver dn> <name>*	Clears a specified Named Password.
-startjob *<job dn>*	Starts the specified job.

Option	Description
-abortjob *<job dn>*	Aborts the specified job.
-getjobrunningstate *<job dn>*	Returns the specified job's running state.
-getjobenabledstate *<job dn>*	Returns the specified job's enabled state.
-getjobnextruntime *<job dn>*	Returns the specified job's next run time.
-updatejob *<job dn>*	Updates the specified job.
-clearallnamedpaswords *<driver dn>*	Clears all Named Passwords set on a specific driver.

If a command line is executed successfully, it returns a zero. If the command line returns anything other than zero, it is an error. For example 0 means success, and -641 means invalid operation. -641 is an eDirectory error code. Table B-9 on page 95 contains other values for specific command line options.

Table B-9 *Command Line Option Values*

Command Line Option	Values
-getstate	0- stopped
	1- starting
	2- running
	3- shutting down
	11- get schema
	Anything else that is returned is an error.
-getstartoption	0- disabled
	1- manual
	2- auto
	Anything else that is returned is an error.
-getcachelimit	0- unlimited
	Anything else that is returned is an error.
-getjobrunningstate	0- stopped
	1- running
	Anything else that is returned is an error.
-getjobenabledstate	0- disabled
	1- enabled
	2- configuration error
	Anything else that is returned is an error.

Command Line Option	Values
-getjobnextruntime	Return is the next scheduled time for the job in eDirectory time format (number of seconds since 00:00:00 Jan 1, 1970 UTC).

Properties of the Driver

C

There are many different fields and values for the driver. Sometimes the information is displayed differently in iManager than in Designer. This section is a reference for all of the fields on the driver as displayed in iManager and Designer.

The information is presented from the viewpoint of iManager. If a field is different in Designer, it is marked with an  icon.

- Section C.1, “Driver Configuration,” on page 97
- Section C.2, “Global Configuration Values,” on page 104
- Section C.3, “Named Passwords,” on page 105
- Section C.4, “Engine Control Values,” on page 106
- Section C.5, “Log Level,” on page 108
- Section C.6, “Driver Image,” on page 109
- Section C.7, “Security Equals,” on page 109
- Section C.8, “Filter,” on page 110
- Section C.9, “Edit Filter XML,” on page 110
- Section C.10, “Misc,” on page 111
- Section C.11, “Excluded Users,” on page 111
- Section C.12, “Driver Manifest,” on page 112
- Section C.13, “Driver Inspector,” on page 112
- Section C.14, “Driver Cache Inspector,” on page 113
- Section C.15, “Inspector,” on page 113
- Section C.16, “Server Variables,” on page 114

C.1 Driver Configuration

In iManager:

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

In Designer:

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

There are different sections under *Driver Configuration.* In this document, each section is listed in a table. The table contains a description of the fields, and the default value or an example of the value that should be specified in the field.

C.1.1 Driver Module

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

In iManager:

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

   See Table C-1 for a list of the driver module options.

In Designer:

1. Open a project in the Modeler, then right-click the driver line and select *Properties* > *Driver Configuration.*
2. Select the *Driver Module* tab.

   See Table C-1 for a list of the driver module options.

Table C-1 *Driver Module Options*

Option	Description
Java	Used to specify the name of the Java class that is instantiated for the shim component of the driver. This class can be located in the `classes` directory as a class file, or in the `lib` directory as a `.jar` file. If this option is selected, the driver is running locally.
Native	Used to specify the name of the `.dll` file that is instantiated for the application shim component of the driver. If this option is selected, the driver is running locally.
Connect to Remote Loader	Used when the driver is connecting remotely to the connected system.
Remote Loader Client Configuration for Documentation	Includes the Remote Loader client configuration information in the driver documentation that is generated by Designer.

C.1.2 Driver Object Password

In iManager:

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

   See Table C-2 for more information.

In Designer:

1. Open a project in the Modeler, then right-click the driver line and click *Properties* > *Driver Configuration*.
2. Click *Driver Module* > *Connect to Remote Loader* > *Driver Object Password* > *Set Password*.

   See Table C-2 for more information.

Table C-2 *Driver Object Password*

Option	Description
Driver Object Password	Use this option to set a password for the driver object. If you are using the Remote Loader, you must enter a password on this page or the remote driver does not run. This password is used by the Remote Loader to authenticate itself to the remote driver shim.

C.1.3 Authentication

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

In iManager:

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

   See Table C-3 for a list of the authentication options.

In Designer:

1. Open a project in the Modeler, then right-click the driver line and select *Properties* > *Driver Configuration*.
2. Click *Authentication*.

   See Table C-3 for a list of the authentication options.

Table C-3 *Authentication Options*

Option	Description
Authentication ID or *User ID*	Specify a user application ID. This ID is used to pass Identity Vault subscription information to the application. Example: `Administrator`
Authentication Context or *Connection Information*	Specify the IP address or name of the server the application shim should communicate with.

Option	Description
Remote Loader Connection Parameters or *Host name* Port *KMO* *Other parameters*	Used only if the driver is connecting to the application through the Remote Loader. The parameter to enter is `hostname=xxx.xxx.xxx.xxx port=xxxx kmo=certificatename`, when the host name is the IP address of the application server running the Remote Loader server and the port is the port the Remote Loader is listening on. The default port for the Remote Loader is 8090. The `kmo` entry is optional. It is only used when there is an SSL connection between the Remote Loader and the Metadirectory engine. Example: `hostname=10.0.0.1 port=8090 kmo=IDMCertificate`
Driver Cache Limit (kilobytes) or *Cache limit (KB)*	Specify the maximum event cache file size (in KB). If it is set to zero, the file size is unlimited. Click *Unlimited* to set the file size to unlimited in Designer.
Application Password or *Set Password*	Specify the password for the user object listed in the *Authentication ID* field.
Remote Loader Password or *Set Password*	Used only if the driver is connecting to the application through the Remote Loader. The password is used to control access to the Remote Loader instance. It must be the same password specified during the configuration of the Remote Loader on the connected system.

C.1.4 Startup Option

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

In iManager:

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

   See Table C-4 for a list of the startup options.

In Designer:

1. Open a project in the Modeler, then right-click the driver line and select *Properties* > *Driver Configuration.*
2. Click *Startup Option*.

   See Table C-4 for a list of the startup options.

Table C-4 *Startup Options*

Option	Description
Auto start	The driver starts every time the Identity Manager server is started.
Manual	The driver does not start when the Identity Manager server is started. The driver must be started through Designer or iManager.
Disabled	The driver has a cache file that stores all of the events. When the driver is set to Disabled, this file is deleted and no new events are stored in the file until the driver state is changed to Manual or Auto Start.
Do not automatically synchronize the driver	This option only applies if the driver is deployed and was previously disabled. If this is not selected, the driver re-synchronizes the next time it is started.

C.1.5 Driver Parameters

In iManager:

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

   See Table C-5 for a list of the driver parameters.

In Designer:

1. Open a project in the Modeler, then right-click the driver line and select *Properties* > *Driver Configuration*.
2. Click *Driver Parameters*.

   See Table C-5 for a list of the driver parameters.

Table C-5 *Driver Parameters*

Parameter	Description
Driver Settings	
<nds>, <input>, <output> Element Handling	Specify *Remove/add elements* if you want the driver shim to remove and add the required XML elements `<nds>`, `<input>`, and `<output>`. The required elements are removed from XML documents sent to the application and are added to XML documents received from the application before presenting the document to the Metadirectory engine. Otherwise, specify *Pass elements through* to turn off this element handling.
Custom Java Extensions	Select *Show* if you have developed custom Java classes to extend the driver shim's functionality. Otherwise, select *Hide*. For more information, see Appendix A, "Using Java Extensions," on page 79.

Parameter	Description
Document Handling	Select *Implemented* if you have developed a custom Java class to process data as XML documents.
Byte array handling	Select *Implemented* if you have developed a custom Java class to process data as a byte array.
Subscriber Transport Layer Replacement	Select *Implemented* if you have developed a custom Java class to replace the default HTTP transport layer for the Subscriber channel.
Publisher Transport Layer Replacement	Select *Implemented* if you have developed a custom Java class to replace the default HTTP transport layer for the Publisher channel.
Schema	Select *Implemented* if you have developed a custom Java class to provide the application schema to the driver.
Subscriber Setting	
URL of the Remote DSML Server	Specify the URL of the remote server and the port number that the server listens on. The URL should begin with http:// unless you have configured SSL settings, in which case it should begin with https:// and use a DNS hostname rather than an IP address.
(Conditional) *Authentication ID*	If the remote server requires an authentication ID, specify the ID in the field. Otherwise, leave the field empty.
(Conditional) *Authentication Password* and *Re-Enter Authentication Password*	Specify the authentication password for the remote server if you specified an *Authentication ID* above. Otherwise, leave the field empty.
Remove Existing Password	Click the box to remove the existing password. Then specify the new password in the *Authentication Password* and *Re-Enter Authentication Password* fields. You cannot change the password without selecting the box.
Truststore File	Specify the name and path of the keystore file containing the trusted certificates used when the remote server is configured to provide server authentication. For example: `c:\security\truststore`. Leave this field empty when server authentication is not used.
Set mutual authentication parameters	Specify *Show* to set mutual authentication information. Specify *Hide* to not use mutual authentication.
Proxy Host and Port	Specify the host address and the host port when a proxy host and port are used. For example: 192.10.1.3:18180. Or, if a proxy host and port are not used, leave this field empty.
Handle HTTP session cookies	Some HTTP applications set cookies and expect them to be present on future requests. Select *Handle Cookies* if you want the driver to keep track of session cookies. Cookies are only kept until the driver is stopped.
Process empty subscriber documents	Indicates whether or not the Subscriber channel should send the empty documents to the target application. Documents could be empty if the policy or the style sheets strip the XML without vetoing the command.

Parameter	Description
Customize HTTP Request Header Fields	Select *Show* to enable customized header fields or select *Hide* to disable the feature. Each of the following fields is conditional, depending on if you select *User* or *Ignore*. ◆ *Authorization*: If you select *Use*, specify the key and value in the appropriate fields. This header is automatically used if you enter an authentication ID and password in the Subscriber Settings. ◆ *Context Type*: If you select *Use*, specify the key and value in the appropriate fields. ◆ *SOAPAction*: If you select *Use*, specify the key and value in the appropriate fields. ◆ *Optional Request Header*: If you select *Use*, specify the key and value in the appropriate fields. You can specify up to three optional request headers.
Publisher Settings	
Listening IP address and port	Specify the IP address of the server where the SOAP driver is installed and the port number that this driver listens on. If you imported a sample configuration file, this field contains the IP address and port that you specified in the wizard.
(Conditional) *Authentication ID*	Specify the *Authentication ID* of the remote server to validate incoming requests. If the remote server does not send an *Authentication ID*, leave this field empty. If you imported a sample configuration file, this field contains the IP address and port that you specified in the wizard.
(Conditional) *Authentication Password* and *Re-Enter Authentication Password*	Specify the authentication password of the remote server to validate incoming requests if you entered an *Authentication ID* above. Otherwise, leave these fields empty.
Remove existing password	Click the box to remove the existing password, then specify the new password in the *Authentication Password* and *Re-Enter Authentication Password* fields. You cannot change the password without selecting the box.
KMO name	Specify the *KMO name* to be used in eDirectory. When the server is configured to accept HTTPS connections, this name becomes the KMO name in eDirectory. The *KMO name* is the name before the “-” (dash) in the RDN. Leave this field empty when a keystore file (see Keystore file below) is used or when HTTPS connections are not used.
Keystore file	Specify the keystore name and path to the keystore file. This file is used when the server is configured to accept HTTPS connections. Leave this field empty when a KMO name is used (see KMO name above) or when HTTPS connections are not used.

Parameter	Description
Keystore password	Specify the keystore file password used with the keystore file specified above when this server is configured to accept HTTPS connections. Leave this field empty when a KMO name is used or when HTTPS connections are not used.
Server key alias	Specify a Server key alias when this server is configured to accept HTTPS connections. Leave this field empty when a KMO name is used or when HTTPS connections are not used.
Server key password	When this server is configured to accept HTTPS connections, this is the key alias password (not the keystore password). Leave this field empty when a KMO name is used (see above) or when HTTPS connections are not used.
Require mutual authentication	When using SSL, it is common to do only server authentication. However, if you want to force both client and server to present certificates during the handshake process, you should require mutual authentication.
Heartbeat Interval in Seconds	Specify the heartbeat interval in seconds. Leave this field empty to turn off the heartbeat. For more information about the heartbeat, see Section 7.8, “Adding a Driver Heartbeat,” on page 53.

C.2 Global Configuration Values

Global configuration values (GCVs) allow you to specify settings for the Identity Manager features such as password synchronization and driver heartbeat, as well as settings that are specific to the function of an individual driver configuration. Some GCVs are provided with the drivers, but you can also add your own.

IMPORTANT: Password Synchronization settings are GCVs, but it’s best to edit them in the graphical interface provided on the Server Variables page for the driver, instead of the GCV page. The Server Variables page that shows password synchronization settings is accessible as a tab like other driver parameters, or by clicking *Password Management* > *Password Synchronization*, searching for the driver, and clicking the driver name. The page contains online help for each password synchronization setting.

In iManager:

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

In Designer:

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

For *Password Configuration*, you should only edit the first two settings listed in Table C-6. The others are GCVs regarding password synchronization that are common to all drivers. They should be edited using iManager in *Passwords* > *Password Synchronization*, not here. Some of them have dependencies on each other that are represented only in the iManager interface. They are explained in "Password Synchronization across Connected Systems" in the *Novell Identity Manager 3.5.1 Administration Guide*.

Table C-6 *Global Configuration Values* > *Password Configuration*

Option	Description
Application accepts passwords from Identity Manager	If *True*, allows passwords to flow from the Identity Manager data store to the connected system.
Identity Manager accepts passwords from application	If *True*, allows passwords to flow from the connected system to Identity Manager.
Publish passwords to NDS password	Use the password from the connected system to set the non-reversible NDS® password in eDirectory.
Publish passwords to Distribution Password	Use the password from the connected system to set the NMAS™ Distribution Password used for Identity Manager password synchronization.
Require password policy validation before publishing passwords	If True, applies NMAS password policies during publish password operations. The password is not written to the data store if it does not comply.
Reset user's external system password to the Identity Manager password on failure	If True, on a publish Distribution Password failure, attempt to reset the password in the connected system using the Distribution Password from the Identity Manager data store.
Notify the user of password synchronization failure via e-mail	If True, notify the user by e-mail of any password synchronization failures.
Connected System or Driver Name	The name of the connected system, application, or Identity Manager driver. This value is used by the e-mail notification templates.

C.3 Named Passwords

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

You can also use the Named Passwords feature to store other pieces of information securely, such as a user name. To configured Named Passwords, see Section 7.7, "Storing Driver Passwords Securely with Named Passwords," on page 46.

In iManager:

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

3 Click *Edit Properties* > *Named Passwords*.

In Designer:

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

C.4 Engine Control Values

The engine control values are a means through which certain default behaviors of the Metadirectory engine can be changed. The values can only be accessed if a server is associated with the Driver Set object.

In iManager:

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

2 Browse to the driver, then click the upper right corner of the driver icon.

3 Click *Edit Properties* > *Engine Control Values*.

See Table C-7 for a list of the engine control values.

In Designer:

1 In the Modeler, right-click the driver line.

2 Select *Properties* > *Engine Control Values*.

3 Click the tooltip icon to the right of the *Engine Control For Server* field. If a server is associated with the Identity Vault, and if you are authenticated, the engine control values display in the large pane.

See Table C-7 for a list of the engine control values.

Table C-7 *Engine Control Values*

Option	Description
Subscriber channel retry interval in seconds	The Subscriber channel retry interval controls how frequently the Metadirectory 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 is presented to the Subscriber channel with type qualifiers. For example, CN=. A True setting means the names are presented in qualified form.

Option	Description
Maximum eDirectory replication wait time in seconds	The maximum eDirectory™ replication wait time controls the maximum time that the Metadirectory engine waits for a particular change to replicate between the local replica and a remote replica. This only affects operations where the Metadirectory 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 Metadirectory engine to a backward-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 in the interest of backward-compatibility with existing DirXML® style sheets that depend on the non-standard behaviors. For example, the behavior of the XPath "!=" operator when one operand is a node set and the other operand is other than a node set is incorrect in DirXML releases up to and including Identity Manager 2.0. This behavior has been corrected; however, the corrected behavior is disabled by default through this control in favor of backward-compatibility with existing DirXML style sheets.
Maximum application objects to migrate at once	This control is used to limit the number of application objects that the Metadirectory engine requests from an application during a single query that is performed as part of a Migrate Objects from Application operation. If java.lang.OutOfMemoryError errors are encountered during a Migrate from Application operation, this number should be set lower than the default. The default is 50. **NOTE:** This control does not limit the number of application objects that can be migrated; it merely limits the batch size.
Set creatorsName on objects created in Identity Vault	This control is used by the Identity Manager engine to determine if the creatorsName attribute should be set to the DN of this driver on all objects created in the Identity Vault by this driver. Setting the creatorsName attribute allows for easily identifying objects created by this driver, but also carries a performance penalty. If not set, the creatorsName attribute defaults to the DN of the NCP™ Server object that is hosting the driver.
Write pending associations	This control determines whether the Identity Manager engine writes a pending association on an object during Subscriber channel processing. Writing a pending association confers little or no benefit but does incur a performance penalty. Nevertheless, the option exists to turn it on for backward compatibility.

Option	Description
Use password event values	This control determines the source of the value reported for the nspmDistributionPassword attribute for Subscriber channel Add and Modify events. Setting the control to False means that the current value of the nspmDistributionPassword is obtained and reported as the value of the attribute event. This means that only the current password value is available. This is the default behavior. Setting the control to True means that the value recorded with the eDirectory event is decrypted and is reported as the value of the attribute event. This means that both the old password value (if it exists) and the replacement password value at the time of the event are available. This is useful for synchronizing passwords to certain applications that require the old password to enable setting a new password.
Enable password synchronization status reporting	This control determines whether the Identity Manager engine reports the status of Subscriber channel password change events. Reporting the status of Subscriber channel password change events allows applications such as the Identity Manager User Application to monitor the synchronization progress of a password change that should be synchronized to the connected application.

C.5 Log Level

Every driver set and 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). Change the log level if you want to track additional message types.

Novell® recommends that you use Novell Audit instead of setting the log levels. See "Integrating Identity Manager with Novell Audit" in the *Identity Manager 3.5.1 Logging and Reporting*.

In iManager:

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

   See Table C-8 for a list of the driver log levels.

In Designer:

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

   See Table C-8 for a list of the driver log levels.

Table C-8 *Driver Log Levels*

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	If selected, 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.

C.6 Driver Image

Allows you to change the image associated with the driver. You can browse 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.

In iManager:

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

In Designer:

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

C.7 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.

In iManager:

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

Designer does not list the users the driver is security equivalent to.

C.8 Filter

Launches the Filter editor. You can edit the filter from this tab.

In iManager:

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

The Filter editor is accessed through the outline view in Designer.

1. In an open project, click the *Outline* tab.
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.

C.9 Edit Filter XML

Allows you to edit the filter directly in XML instead of using the Filter Editor.

In iManager:

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

You can edit the Filter in XML through the Filter Editor.

1. In an open project, click the *Outline* tab.
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, then click *XML Source* at the bottom of the Filter Editor.

C.10 Misc

Allows you to add a trace level to your driver. With the trace level set, DSTrace displays the Identity Manager events as the Metadirectory engine processes the events. The trace level only affects the driver it is set for. Use the trace level for troubleshooting issues with the driver when the driver is deployed. DSTrace displays the output of the specified trace level.

In iManager:

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

   See Table C-9 for a list of the driver trace levels.

In Designer:

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

   See Table C-9 for a list of the driver trace levels.

Table C-9 *Driver Trace Levels*

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. As long as the file is specified, Java information is written to this file. If you do not need to debug Java, leave this field blank.
Trace file size limit	Allows you to set a limit for the Java trace file. If you set the file size to Unlimited, the file grows in size until there is no disk space left. **NOTE:** The trace file is created in multiple files.Identity Manager automatically divides the maximum file size by ten and creates ten separate files. The combined size of these files equals the maximum trace file size.
Trace name	Driver trace messages are prepended with the value entered in this field.
Use setting from Driver Set	This option is only available in Designer. It allows the driver to use the same setting that is set on the Driver Set object.

C.11 Excluded Users

Use this page to create a list of users or resources that are not replicated to the application. Novell recommends that you add all objects that represent an administrative role to this list (for example, the Admin object).

In iManager:

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

Designer does not list the excluded users.

C.12 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.

In iManager:

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

In Designer:

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

C.13 Driver Inspector

The Driver Inspector page displays information about all of the objects associated with the driver.

- **Driver:** A link to run the *Driver Overview* on the driver that is being inspected.
- **Driver Set:** A link to run the *Driver Set Overview* of the driver set that holds the driver.
- **Delete:** Deletes the associations of the selected objects.
- **Refresh:** Select this option to re-read all of the objects associated with the driver and refresh the displayed information.
- **Actions:** Allows you to perform actions on the objects associated with the driver. Click *Actions* to expand the menu, which includes:
  - **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.

C.14 Driver Cache Inspector

The Driver Cache Inspector page uses a table format to display information about the cache file that stores events while the driver is stopped.

- **Driver:** A link to run the *Driver Overview* on the driver that is associated with this cache file.
- **Driver Set:** A link to run the *Driver Set Overview* on the driver set that holds the driver.
- **Driver's cache on:** Lists the server object that contains this instance of the cache file.
- **Start/Stop Driver icons:** Displays the current state of the driver and allows you to start or stop the driver.
- **Edit icon:** Allows you to edit the properties of the currently selected Server object.
- **Delete:** Deletes the selected items from the cache file.
- **Refresh:** Select this option to re-read the cache file and refresh the displayed information.
- **Show:** Limits the number of items to be displayed. The options are:
  - 25 per page
  - 50 per page
  - 100 per page
  - Other: Allows you to specify a desired number.
- **Actions:** Allows you to perform actions on the entries in the cache file. Click *Actions* to expand the menu, which includes:
  - **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.

C.15 Inspector

The Inspector displays information about the connected system without directly accessing the system. Designer does not have this option.

C.16 Server Variables

This page lets you enable and disable password synchronization and the associated options for the selected driver.

When setting up password synchronization, consider both the settings on this page for an individual driver and the Universal Password Configuration options in your password policies.

This page lets you control which password Identity Manager updates directly, either the Universal Password for an Identity Vault, or the Distribution Password used for password synchronization by Identity Manager.

However, Novell Modular Authentication Service (NMAS™) controls whether the various passwords inside the Identity Vault are synchronized with each other. Password Policies are enforced by NMAS, and they include settings for synchronizing Universal Password, NDS® Password, Distribution Password, and Simple Password.

To change these settings in iManager:

1. In iManager, select *Passwords* > *Password Policies*.
2. Select a password policy, then click *Edit*.
3. Select *Universal Password*.

   This option is available from a drop-down list or a tab, depending on your version of iManager and your browser.
4. Select *Configuration Options*, make changes, then click *OK*.

NOTE: Enabling or disabling options on this page corresponds to values of True or False for certain global configuration values (GCVs) used for password synchronization in the driver parameters. Novell recommends that you edit them here in the graphical interface, instead of on the GCVs page. This interface helps ensure that you don't set conflicting values for the password synchronization GCVs.

Option	Description
Identity Manager accepts password (Publisher Channel)	If this option is enabled, Identity Manager allows passwords to flow from the connected system driver into the Identity Vault data store. Disabling this option means that no <*password*> elements are allowed to flow to Identity Manager. They are stripped out of the XML by a password synchronization policy on the Publisher channel. If this option is enabled, and the option below it for Distribution Password is disabled, a <*password*> value coming from the connected system is written directly to the Universal Password in the Identity Vault if it is enabled for the user. If the user's password policy does not enable Universal Password, the password is written to the NDS Password.

Option	Description
Use Distribution Password for password synchronization	To use this setting, you must have a version of eDirectory that supports Universal Password, regardless of whether you have enabled Universal Password in your password policies. If this option is enabled, a password value coming from the connected system is written to the Distribution Password. The Distribution Password is reversible, which means that it can be retrieved from the Identity Vault data store for password synchronization. It is used by Identity Manager for bidirectional password synchronization with connected systems. For Identity Manager to distribute passwords to connected systems, this option must be enabled. NMAS and Password policies control whether the Distribution Password is synchronized with other passwords in the Identity Vault. By default, the Distribution Password is the same as the Universal Password in the Identity Vault. If the password in the Identity Vault is to be independent of password synchronization, so that Identity Manager is a conduit only for synchronizing passwords among connected systems, change this default setting. In the Universal Password Configuration Options in a Password policy, disable *Synchronize Universal Password with Distribution Password*. This use of Identity Manager password synchronization is also referred to as “tunneling.”
Accept password only if it complies with user’s Password Policy	To use this setting, users must have a Password policy assigned that has Universal Password enabled, and Advanced Password Rules enabled and configured. If this option is chosen, Identity Manager does not write a password from this connected system to the Distribution Password in the Identity Manager data store or publish it to connected systems unless the password complies with the user's Password policy. By using the notification option that is also on this page, you can inform users when a password is not set because it is not compliant.

Option	Description
If password does not comply, ignore Password Policy on the connected system by resetting user's password to the Distribution Password	This option lets you enforce Password policies on the connected system by replacing a password that does not comply. If you select this option, and a user's password on the connected system does not comply with the user's Password policy, Identity Manager resets the password on the connected system by using the Distribution Password from the Identity Vault data store. Keep in mind that if you do not select this option, user passwords can become out-of-sync on connected systems. By using the notification option that is also on this page, you can inform users when a password is not set or reset. Notification is especially helpful for this option. If the user changes to a password that is allowed by the connected system but rejected by Identity Manager because of the Password policy, the user won't know that the password has been reset until the user receives a notification or tries to log in to the connected system with the old password. **NOTE:** Consider the connected system's password policies when deciding whether to use this option. Some connected systems might not allow the reset because they don't allow you to repeat passwords.
Always accept password; ignore Password Policies	If you select this option, Identity Manager does not enforce the user's Password policy for this connected system. Identity Manager writes the password from this connected system to the Distribution Password in the Identity Vault data store, and distributes it to other connected systems, even if the password does not comply with the user's Password policy.
Application accepts passwords (Subscriber Channel)	If you select this option, the driver sends passwords from the Identity Vault data store to this connected system. This also means that if a user changes the password on a different connected system that is publishing passwords to the Distribution Password in the Identity Vault data store, the password is changed on this connected system. By default, the Distribution Password is the same as the Universal Password in the Identity Vault, so changes to the Universal Password made in the Identity Vault are also sent to the connected system. If you want the password in the Identity Vault to be independent of password synchronization, so that Identity Manager is a conduit only for synchronizing passwords among connected systems, you can change this default setting. In the Universal Password Configuration Options in a password policy, disable *Synchronize Universal Password with Distribution Password*. This use of password synchronization is also referred to as "tunneling."
Notify the user of password synchronization failure via-email	If you select this option, e-mail is sent to the user if a password is not synchronized, set, or reset. The e-mail that is sent to the user is based on an e-mail template. This template is provided by the password synchronization application. However, for the template to work, you must customize it and specify an e-mail server to send the notification messages. **NOTE:** To set up e-mail notification, select *Passwords* > *Edit EMail Templates*.