AUTHORIZED DOCUMENTATION

Driver for SOAP Implementation Guide

Novell®
Identity Manager

3.6

July 23, 2008

www.novell.com

Legal Notices

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

Third-Party Materials

Contents

About This Guide

This guide explains how to install and configure the Identity Manager 3.6 Driver for SOAP (also called the SOAP driver). The guide includes the following information:

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 the Identity Manager 3.6 Drivers Documentation Web site (http://www.novell.com/documentation/idm36drivers/index.html).

Additional Documentation

For information on Identity Manager, see the Identity Manager Documentation Web site (http://www.novell.com/documentation/idm36).

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 SOAP driver:

- Section 1.1, "Driver Concepts," on page 9
- Section 1.2, "Support for Standard Driver Features," on page 11

1.1 Driver Concepts

This section contains the following information:

- "Data Management" on page 9
- "How the Driver Works" on page 10

1.1.1 Data Management

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

- "SOAP" on page 9
- "SPML and DSML" on page 10
- "XML" on page 10
- "HTTP" on page 10

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 Section 5.1, “Understanding the DSML Configuration,” on page 25 and Section 5.2, “Understanding the SPML Configuration,” on page 25.

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

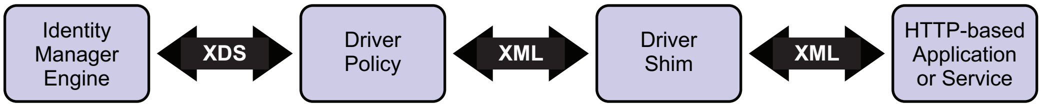
HTTP is a protocol used to request and transmit data over the Internet or other computer network. The protocol works well in an Internet infrastructure and with firewalls.

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.

1.1.2 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.

1.1.3 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, “Managing Operation Data,” on page 27.

1.2 Support for Standard Driver Features

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

- Section 1.2.1, “Local Platforms,” on page 12
- Section 1.2.2, “Remote Platforms,” on page 12
- Section 1.2.3, “Entitlements,” on page 12
- Section 1.2.4, “Password Synchronization Support,” on page 12
- Section 1.2.5, “Information Synchronized,” on page 12

1.2.1 Local Platforms

A local installation is an installation of the driver on the Metadirectory server. The SOAP driver can be installed on the operating systems supported for the Metadirectory server.

For information about the operating systems supported for the Metadirectory server, see "Metadirectory Server" in "System Requirements" in the *Identity Manager 3.6 Installation Guide*.

1.2.2 Remote Platforms

The SOAP driver can use the Remote Loader service to run on a server other than the Metadirectory server. The SOAP driver can be installed on the operating systems supported for the Remote Loader.

For information about the supported operating systems, see "Remote Loader" in "System Requirements" in the *Identity Manager 3.6 Installation Guide*.

1.2.3 Entitlements

The SOAP driver does not have entitlement functionality defined in the basic configuration files provided as examples. However, the driver does support entitlements, if there are policies created for the driver to consume.

1.2.4 Password Synchronization Support

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

1.2.5 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.
- 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 B, "Using Java Extensions," on page 49.

Installing the Driver Files

2

You must install the SOAP driver on a server that has HTTP access to the Web service with which the driver will communicate. This can be an existing Metadirectory server or a non-Metadirectory server that meets the system requirements for running the Remote Loader service (see see "Remote Loader" in "System Requirements" in the *Identity Manager 3.6 Installation Guide*).

By default, the SOAP driver files are installed on the Metadirectory server at the same time as the Metadirectory engine. The installation program extends the Identity Vault's schema and installs both the driver shim and the driver configuration files. It does not create the driver in the Identity Vault (see Chapter 3, "Creating a New Driver," on page 15) or upgrade an existing driver's configuration (see Chapter 4, "Upgrading an Existing Driver," on page 23).

If the SOAP driver files are not currently located on the server where you want to run the driver:

- Install the driver files on an existing Metadirectory server, using the instructions in "Installing the Metadirectory Server" in the *Identity Manager 3.6 Installation Guide*.
- Install the Remote Loader (required to run the driver on a non-Metadirectory server) and the driver files on a non-Metadirectory server where you want to run the driver. See "Installing the Remote Loader" in the *Identity Manager 3.6 Installation Guide*.

Creating a New Driver

3

After the SOAP driver files are installed on the server where you want to run the driver (see Chapter 2, "Installing the Driver Files," on page 13), you can create the driver in the Identity Vault. You do so by importing the basic driver configuration file and then modifying the driver configuration to suit your environment.

The SOAP driver comes with a basic configuration file for SPML protocol and another one for DSML protocol. For information about the two protocols, see "SPML and DSML" on page 10, Section 5.1, "Understanding the DSML Configuration," on page 25, and Section 5.2, "Understanding the SPML Configuration," on page 25.

The following sections provide instructions to create the driver:

- Section 3.1, "Creating the Driver in Designer," on page 15
- Section 3.2, "Creating the Driver in iManager," on page 18
- Section 3.3, "Activating the Driver," on page 21

3.1 Creating the Driver in Designer

You create the SOAP driver by importing the driver's basic configuration file and then modifying the configuration to suit your environment. After you've created and configured the driver, you need to deploy it to the Identity Vault and start it.

- Section 3.1.1, "Importing the Driver Configuration File," on page 15
- Section 3.1.2, "Configuring the Driver," on page 17
- Section 3.1.3, "Deploying the Driver," on page 17
- Section 3.1.4, "Starting the Driver," on page 18

3.1.1 Importing the Driver Configuration File

1. In Designer, open your project.
2. In the Modeler, right-click the driver set where you want to create the driver, then select *New* > *Driver* to display the Driver Configuration Wizard.
3. In the Driver Configuration list, select *SOAP-SPML* or *SOAP-DSML*, then click *Run*.
4. On the Import Information Requested page, fill in the following fields:

   Driver Name: Specify a name that is unique within the driver set.

   URL of the Remote SPML Provisioning Service Point: This field applies only if you selected the SOAP-SPML configuration.

   Specify the URL for the Provisioning Service Point (PSP) that will listen for, process, and return the results of the SPML requests generated by the SOAP driver.

   URL of the Remote DSML Server: This field applies only if you selected the SOAP-DSML configuration.

   Specify the URL for the DSML Server that will listen for, process, and return the results of the DSML requests generated by the SOAP driver.

User Container: This field applies only if you selected the SOAP-SPML configuration.

Select the Identity Vault container where users are stored. This value becomes the default for all drivers in the driver set. If you don't want to change this value for all drivers, leave the current value in this field and change the value on the driver's Global Configuration Values page after you've finished importing the driver.

Driver is Local/Remote: Select *Local* if this driver will run on the Metadirectory server without using the Remote Loader service. Select *Remote* if you want the driver to use the Remote Loader service, either locally on the Metadirectory server or remotely on another server.

Subsciber Keystore Password: This field applies only if you selected the SOAP-DSML configuration. It is used to configure a secure HTTPS connection for the Subscriber channel. Unless you already have a keystore and certificates set up on the driver server and the DSML server, leave this field empty. Complete instructions for creating a secure connection are provided in Chapter 6, "Securing Communication," on page 31.

Publisher Keystore Password: This field applies only if you selected the SOAP-DSML configuration. It is used to configure a secure HTTPS connection for the Publisher channel. Unless you already have a keystore and certificates set up on the driver server and the DSML server, leave this field empty. Complete instructions for creating a secure connection are provided in Chapter 6, "Securing Communication," on page 31.

Publisher Server Key Password: This field applies only if you selected the SOAP-DSML configuration. It is used to configure a secure HTTPS connection for the Publisher channel. Unless you already have a keystore and certificates set up on the driver server and the DSML server, leave this field empty. Complete instructions for creating a secure connection are provided in Chapter 6, "Securing Communication," on page 31.

5 (Conditional) If you chose to run the driver remotely, click *Next*, then fill in the fields listed below. Otherwise, skip to Step 6.

   Remote Host Name and Port: Specify the hostname or IP address of the server where the driver's Remote Loader service is running.

   Driver Password: Specify the driver object password that is defined in the Remote Loader service. The Remote Loader requires this password to authenticate to the Metadirectory server.

   Remote Password: Specify the Remote Loader's password (as defined on the Remote Loader service). The Metadirectory engine (or Remote Loader shim) requires this password to authenticate to the Remote Loader

6 Click *Next* to import the driver configuration.

   At this point, the driver is created from the basic configuration file. To ensure that the driver works the way you want it to for your environment, you must review and modify the driver's default configuration settings.

7 To review or modify the default configuration settings, click *Configure*, then continue with the next section, Configuring the Driver.

   or

   To skip the configuration settings at this time, click *Close*. When you are ready to configure the settings, continue with Configuring the Driver.

3.1.2 Configuring the Driver

After importing the driver configuration file, you need to configure the driver before it can run. You should complete the following tasks to configure the driver:

- **Configure the driver parameters:** There are many settings that can help you customize and optimize the driver. The settings are divided into categories such as Driver Configuration, Engine Control Values, and Global Configuration Values (GCVs). Although it is important for you to understand all of the settings, your first priority should be to review the Driver Parameters located on the Driver Configuration page. The Driver Parameters let you configure the LDAP directory type, publication method, and other parameters associated with the Publisher channel.
- **Customize the driver policies and filter:** The driver policies and filter control data flow between the Identity Vault and the application. You should ensure that the policies and filters reflect your business needs. For instructions, see Chapter 5, “Customizing the Driver,” on page 25.
- **Set Up a Secure HTTPS Connection:** The connection between the driver and the SPML or DSML server can be configure to use a secure HTTPS connection rather than an HTTP connection. For instructions, see Chapter 6, “Securing Communication,” on page 31

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

3.1.3 Deploying the Driver

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

1. In Designer, open your project.
2. In the Modeler, right-click the driver icon or the driver line, then select *Live* > *Deploy*.
3. If you are authenticated to the Identity Vault, skip to Step 5; otherwise, specify the following information:
   - **Host:** Specify the IP address or DNS name of the server hosting the Identity Vault.
   - **Username:** Specify the DN of the user object used to authenticate to the Identity Vault.
   - **Password:** Specify the user’s password.
4. Click *OK*.
5. Read the deployment summary, then click *Deploy*.
6. Read the message, then click *OK*.
7. Click *Define Security Equivalence* to assign rights to the driver.

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

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

   7b Click *OK* twice.
8. Click *Exclude Administrative Roles* to exclude users that should not be synchronized.

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

  8a Click *Add*, then browse to and select the user object you want to exclude.

  8b Click *OK*.

  8c Repeat Step 8a and Step 8b for each object you want to exclude.

  8d Click *OK*.

9 Click *OK*.

10 Continue with the next section, Starting the Driver.

3.1.4 Starting the Driver

When a driver is created, it is stopped by default. To make the driver work, you must start the driver and cause events to occur. Identity Manager is an event-driven system, so after the driver is started, it won't do anything until an event occurs.

To start the driver:

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

3.2 Creating the Driver in iManager

You create the GroupWise driver by importing the driver's basic configuration file and then modifying the configuration to suit your environment. After you've created and configured the driver, you need to start it.

- Section 3.2.1, "Importing the Driver Configuration File," on page 18
- Section 3.2.2, "Configuring the Driver," on page 21
- Section 3.2.3, "Starting the Driver," on page 21

3.2.1 Importing the Driver Configuration File

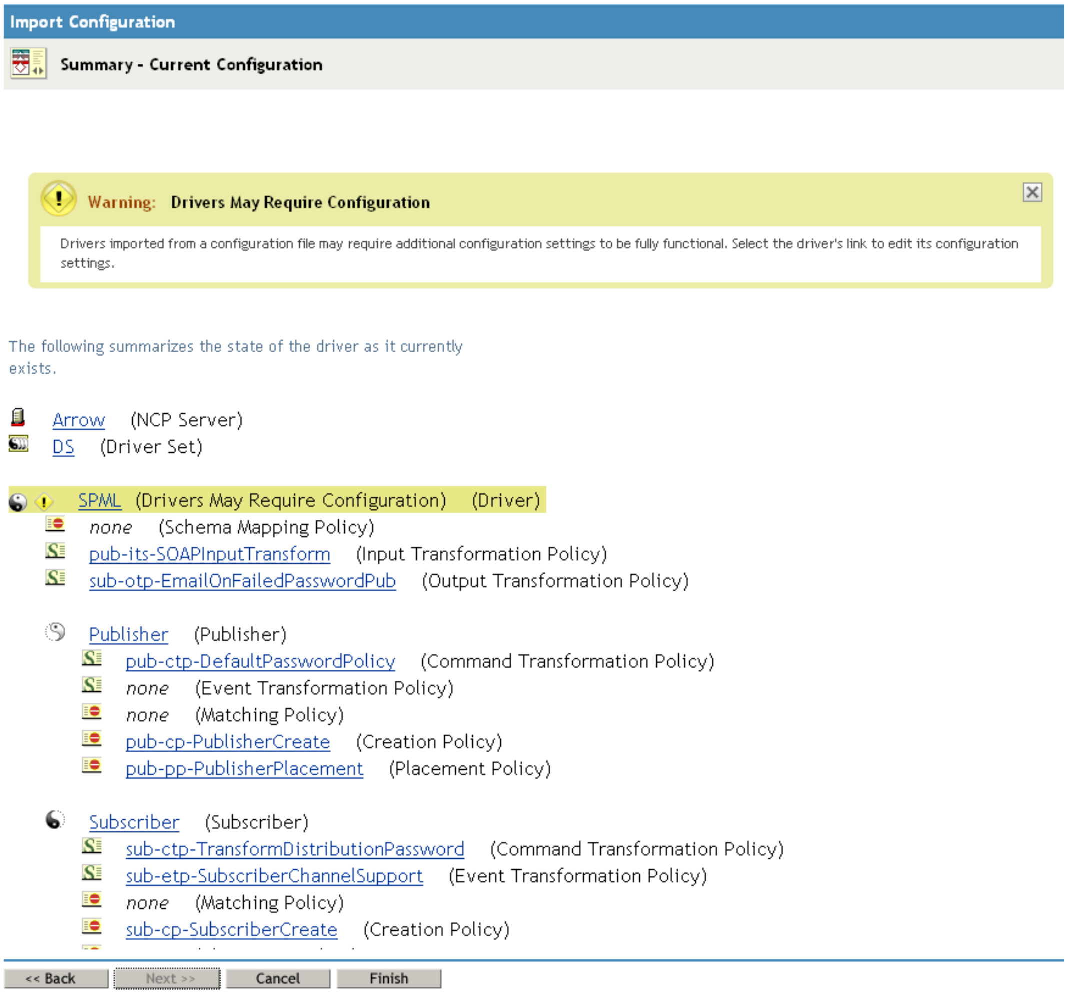
1. In iManager, click to display the Identity Manager Administration page.
2. In the Administration list, click *Import Configuration* to launch the Import Configuration Wizard.
3. Follow the wizard prompts, filling in the requested information (described below) until you reach the Summary page.

Prompt	Description
Where do you want to place the new driver?	You can add the driver to an existing driver set, or you can create a new driver set and add the driver to the new set. If you choose to create a new driver set, you are prompted to specify the name, context, and server for the driver set.

Prompt	Description
Import a configuration into this driver set	Use the default option, *Import a configuration from the server (.XML file)*. In the *Show* field, select *Identity Manager 3.6 configurations*. In the *Configurations* field, select the SOAP-SPML or SOAP-DSML file.
Driver name	Type a name for the driver. The name must be unique within the driver set.
URL of the Remote SPML Provisioning Service Point	This field applies only if you selected the SOAP-SPML configuration. Specify the URL for the Provisioning Service Point (PSP) that will listen for, process, and return the results of the SPML requests generated by the SOAP driver.
URL of the Remote DSML Server	This field applies only if you selected the SOAP-DSML configuration. Specify the URL for the DSML server that will listen for, process, and return the results of the SPML requests generated by the SOAP driver.
User Container	This field applies only if you selected the SOAP-SPML configuration. Select the Identity Vault container where users are stored. This value becomes the default for all drivers in the driver set. If you don't want to change this value for all drivers, leave the current value in this field and change the value on the driver's Global Configuration Values page after you've finished importing the driver.
Driver is Local/Remote	Select *Local* if this driver will run on the Metadirectory server without using the Remote Loader service. Select *Remote* if you want the driver to use the Remote Loader service, either locally on the Metadirectory server or remotely on another server.
Remote Host Name and Port	This applies only if the driver is running remotely. Specify the host name or IP address of the server where the driver's Remote Loader service is running.
Driver Password	This applies only if the driver is running remotely. Specify the driver object password that is defined in the Remote Loader service. The Remote Loader requires this password to authenticate to the Metadirectory server.
Remote Password	This applies only if the driver is running remotely. Specify the Remote Loader's password (as defined on the Remote Loader service). The Metadirectory engine (or Remote Loader shim) requires this password to authenticate to the Remote Loader

Prompt	Description
Define Security Equivalences	The driver requires rights to objects within the Identity Vault and to the input and output directories on the server. The Admin user object is most often used to supply these rights. However, you might want to create a DriversUser (for example) and assign security equivalence to that user. Whatever rights that the driver needs to have on the server, the DriversUser object must have the same security rights.
Exclude Administrative Roles	You should exclude any administrative User objects (for example, Admin and DriversUser) from synchronization.

When you finish providing the information required by the wizard, a Summary page, similar to the following is displayed.



Warning: **Drivers May Require Configuration**

Drivers imported from a configuration file may require additional configuration settings to be fully functional. Select the driver's link to edit its configuration settings.

The following summarizes the state of the driver as it currently exists.

Arrow (NCP Server)
DS (Driver Set)

SPML (Drivers May Require Configuration) (Driver)
- *none* (Schema Mapping Policy)
- pub-its-SOAPInputTransform (Input Transformation Policy)
- sub-otp-EmailOnFailedPasswordPub (Output Transformation Policy)

- Publisher (Publisher)
  - pub-ctp-DefaultPasswordPolicy (Command Transformation Policy)
  - *none* (Event Transformation Policy)
  - *none* (Matching Policy)
  - pub-cp-PublisherCreate (Creation Policy)
  - pub-pp-PublisherPlacement (Placement Policy)

- Subscriber (Subscriber)
  - sub-ctp-TransformDistributionPassword (Command Transformation Policy)
  - sub-etp-SubscriberChannelSupport (Event Transformation Policy)
  - *none* (Matching Policy)
  - sub-cp-SubscriberCreate (Creation Policy)

At this point, the driver is created from the basic configuration file. To ensure that the driver works the way you want it to for your environment, you must review and modify the driver's default configuration settings.

4 To modify the default configuration settings, click the linked driver name, then continue with the next section, Configuring the Driver.

or

To skip the configuration settings at this time, click *Finish*. When you are ready to configure the settings, continue with Configuring the Driver.

3.2.2 Configuring the Driver

After importing the driver configuration file, you need to configure the driver before it can run. You should complete the following tasks to configure the driver:

- **Configure the driver parameters:** There are many settings that can help you customize and optimize the driver. The settings are divided into categories such as Driver Configuration, Engine Control Values, and Global Configuration Values (GCVs). Although it is important for you to understand all of the settings, your first priority should be to review the Driver Parameters located on the Driver Configuration page. The Driver Parameters let you configure the LDAP directory type, publication method, and other parameters associated with the Publisher channel.
- **Customize the driver policies and filter:** The driver policies and filter control data flow between the Identity Vault and the application. You should ensure that the policies and filters reflect your business needs. For instructions, see Chapter 5, "Customizing the Driver," on page 25.
- **Set Up a Secure HTTPS Connection:** The connection between the driver and the SPML or DSML server can be configure to use a secure HTTPS connection rather than an HTTP connection. For instructions, see Chapter 6, "Securing Communication," on page 31

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

3.2.3 Starting the Driver

When a driver is created, it is stopped by default. To make the driver work, you must start the driver and cause events to occur. Identity Manager is an event-driven system, so after the driver is started, it won't do anything until an event occurs.

To start the driver:

1. In iManager, click to display the Identity Manager Administration page.
2. Click *Identity Manager Overview*.
3. Browse to and select the driver set object that contains the driver you want to start.
4. Click the driver set name to access the Driver Set Overview page.
5. Click the upper right corner of the driver, then click *Start driver*.

3.3 Activating the Driver

If you created the driver in a driver set where you've already activated the Metadirectory engine and service drivers, the driver inherits the activation. If you created the driver in a driver set that has not been activated, you must activate the driver within 90 days. Otherwise, the driver stops working.

For information on activation, refer to "Activating Novell Identity Manager Products" in the *Identity Manager 3.6 Installation Guide*.

Upgrading an Existing Driver

4

The following sections provide information to help you upgrade an existing driver to version 3.6:

- Section 4.1, "Supported Upgrade Paths," on page 23
- Section 4.2, "What's New in Version 3.6," on page 23
- Section 4.3, "Upgrade Procedure," on page 23

4.1 Supported Upgrade Paths

You can upgrade from any 3.*x* version of the SOAP driver. Upgrading a pre-3.*x* version of the driver directly to version 3.6 is not supported.

4.2 What's New in Version 3.6

Version 3.6 of the driver adds the ability to set custom HTTP request headers using operation data. See "Overriding the Authorization Header" on page 29.

4.3 Upgrade Procedure

The process for upgrading the SOAP driver is the same as for other Identity Manager drivers. For detailed instructions, see the *Identity Manager 3.6 Installation Guide*.

Customizing the Driver

5

The following sections provide information to help you understand what the driver does and what customization you might need to make to the driver:

- Section 5.1, "Understanding the DSML Configuration," on page 25
- Section 5.2, "Understanding the SPML Configuration," on page 25
- Section 5.3, "Handling Modify Events for Unassociated Objects on the Publisher Channel," on page 26
- Section 5.4, "Creating XSLT Style Sheets," on page 26
- Section 5.5, "Managing Operation Data," on page 27

5.1 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 customization.
- 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.

5.2 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.

5.3 Handling Modify Events for Unassociated Objects on the Publisher Channel

The Publisher channel of the 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 successfully 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 acceptable, 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.

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 3.6*.

5.5 Managing 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:

- 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.
- Override default Subscriber options on a per-command basis.

As discussed in Chapter 1, "Overview," on page 9, 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 27
- Section 5.5.2, "Using Operation Data to Override Default Subscriber Options," on page 27

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 they are 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 three ways to override default Subscriber options for a command:

- "Creating and Using Multiple Subscriber Option Sets (connections)" on page 28
- "Overriding Single Subscriber Options" on page 28
- "Overriding the Authorization Header" on page 29

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 the “SOAPAction” key	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>
```

Overriding the Authorization Header

You can set the Authorization header dynamically (from within policy) in `<operation data>`.

Example:

```
<operation-data>

<request-headers remove-existing="true">

<request-header name="Authorization">Basic cn=admin,o=n:n</
request-header>

<request-header name="SOAPAction">#batchRequest</request-header>

</request-headers>

</operation-data>
```

The `remove-existing` flag defines whether the set of request-headers defined in the subscriber-options should be used in addition to the new headers defined in operation-data. If the Authorization header already exists, it is overridden. Otherwise, it is added as new.

Securing Communication

6

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

The following sections provide instructions for creating a secure connection:

- Section 6.1, "Configuring the Publisher Channel," on page 31
- Section 6.2, "Configuring the Subscriber Channel," on page 32

6.1 Configuring the Publisher Channel

1 Create a server certificate in iManager.

  1a In the *Roles and Tasks* view, 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 In the *Roles and Tasks* view, click *Directory Administration > Modify Object*.

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

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

  2c Click *Certificate > Self Signed Certificate*.

  2d Click *Export*.

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

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

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

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

  2h Click *Save* 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 Enter the following command at a command prompt:

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

For example:

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

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

- **4a** In iManager, in the *Roles and Tasks* view, 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*.

6.2 Configuring the Subscriber Channel

The Subscriber channel sends information from the Identity Vault to the Web service. To establish a secure connection for the Subscriber channel, you need a trust store containing a certificate issued by the certificate authority that signed the server's certificate. See Section 6.1, "Configuring the Publisher Channel," on page 31 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, in the *Roles and Tasks* view, 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 *Keystore File* setting, specify the path to the trust store you created in Step 1.

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

Managing the Driver

7

As you work with the SOAP driver, there are a variety of management tasks you might need to perform, including the following:

- Starting, stopping, and restarting the driver
- Viewing driver version information
- Using Named Passwords to securely store passwords associated with the driver
- Monitoring the driver's health status
- Backing up the driver
- Inspecting the driver's cache files
- Viewing the driver's statistics
- Using the DirXML® Command Line utility to perform management tasks through scripts
- Securing the driver and its information

Because these tasks, as well as several others, are common to all Identity Manager drivers, they are included in one reference, the *Identity Manager 3.6 Common Driver Administration Guide*.

Troubleshooting the Driver

8

You can log Identity Manager events by 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 the *Identity Manager 3.6 Integration Guide for Novell Audit*.

This section contains the following information on error messages:

- Section 8.1, "Driver Shim Errors," on page 35
- Section 8.2, "Java Customization Errors," on page 38
- Section 8.3, "Troubleshooting Driver Processes," on page 39

8.1 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 the 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 Table A-6, "Subscriber Settings," on page 45 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, "Customizing the Driver," on page 25 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 Table A-7, "Publisher Settings," on page 46 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 Table A-7, "Publisher Settings," on page 46 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 Table A-7, "Publisher Settings," on page 46 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

8.2 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 that 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

8.3 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. For more information, see "Viewing Identity Manager Processes" in the *Identity Manager 3.6 Common Driver Administration Guide*.

Driver Properties

A

This section provides information about the Driver Configuration and Global Configuration Values properties for the SOAP driver. These are the only unique properties for drivers. All other driver properties (Named Password, Engine Control Values, Log Level, and so forth) are common to all drivers. Refer to "Driver Properties" in the *Identity Manager 3.6 Common Driver Administration Guide* for information about the common properties.

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

- Section A.1, "Driver Configuration," on page 41
- Section A.2, "Global Configuration Values," on page 47

A.1 Driver Configuration

In iManager:

1 Click to display the Identity Manager Administration page.

2 Open the driver set that contains the driver whose properties you want to edit:

  2a In the *Administration* list, click *Identity Manager Overview*.

  2b If the driver set is not listed on the *Driver Sets* tab, use the *Search In* field to search for and display the driver set.

  2c Click the driver set to open the Driver Set Overview page.

3 Locate the driver icon, then click the upper right corner of the driver icon to display the *Actions* menu.

4 Click *Edit Properties* to display the driver's properties page.

By default, the Driver Configuration page is displayed.

In Designer:

1 Open a project in the Modeler.

2 Right-click the driver icon or line, then select click *Properties > Driver Configuration.*

The Driver Configuration options are divided into the following sections:

A.1.1 Driver Module

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

Table A-1 *Driver Module*

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. The Java class name is: `com.novell.nds.dirxml.driver.soap.SOAPDriver`
Native	This option is not used with the SOAP driver.
Connect to Remote Loader	Used when the driver is connecting remotely to the connected system. Designer includes two suboptions: ◆ *Driver Object Password*: Specifies a password for the Driver object. If you are using the Remote Loader, you must enter a password on this page. Otherwise, the remote driver does not run. The Remote Loader uses this password to authenticate itself to the remote driver shim. ◆ *Remote Loader Client Configuration for Documentation*: Includes information on the Remote Loader client configuration when Designer generates documentation for the driver.

A.1.2 Driver Object Password (iManager Only)

Table A-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.

A.1.3 Authentication

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

Table A-3 *Authentication*

Option	Description
Authentication ID or *User ID*	This option is not used with the SOAP driver. The SOAP driver requires separate authentication settings for both the Publisher channel and the Subscriber channel.

Option	Description
Authentication Context or *Connection Information*	This option is not used with the SOAP driver.
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*	This option is not used with the SOAP driver.
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.

A.1.4 Startup Option

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

Table A-4 *Startup Option*

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.

Option	Description
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.

A.1.5 Driver Parameters

The Driver Parameters section lets you configure the driver-specific parameters. When you change driver parameters, you tune driver behavior to align with your network environment.

The parameters are presented by category:

- Table A-5, "Driver Settings," on page 44
- Table A-6, "Subscriber Settings," on page 45
- Table A-7, "Publisher Settings," on page 46

Table A-5 *Driver Settings*

Option	Description
<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 B, "Using Java Extensions," on page 49.
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.

Table A-6 *Subscriber Settings*

Option	Description
URL of the Remote DSML Server or *URL of the Remote SPML Provisioning Service Point*	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.
Authentication Password	Specify the authentication password for the remote server if you specified an *Authentication ID* above. Otherwise, leave the field empty. If you need to clear the password, select *Remove existing password*, then click *Apply*.
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.
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.

***Table A-7** Publisher Settings*

Option	Description
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.
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.
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. If you need to clear the password, select *Remove existing password*, then click *Apply*.
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.
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 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.

A.2 Global Configuration Values

Global configuration values (GCVs) are values that can be used by the driver to control functionality. GCVs are defined on the driver or on the driver set. Driver set GCVs can be used by all drivers in the driver set. Driver GCVs can be used only by the driver on which they are defined.

The SOAP driver includes several predefined GCVs. You can also add your own if you discover you need additional ones as you implement policies in the driver.

To access the driver's GCVs in iManager:

1 Click to display the Identity Manager Administration page.

2 Open the driver set that contains the driver whose properties you want to edit.

  2a In the *Administration* list, click *Identity Manager Overview*.

  2b If the driver set is not listed on the *Driver Sets* tab, use the *Search In* field to search for and display the driver set.

  2c Click the driver set to open the Driver Set Overview page.

3 Locate the driver icon, click the upper right corner of the driver icon to display the *Actions* menu, then click *Edit Properties*.

or

To add a GCV to the driver set, click *Driver Set*, then click *Edit Driver Set properties*.

To access the driver's GCVs in Designer:

1 Open a project in the Modeler.

2 Right-click the driver icon or line, then select *Properties > Global Configuration Values.*

or

To add a GCV to the driver set, right-click the driver set icon , then click *Properties > GCVs*.

The global configuration values are organized as follows:

Table A-8 *General Values*

Option	**Description**
SPML Identifier Type	This option is only available when using the SPML configuration for the driver. SPML introduces the concept of a Provisioning Service Target Data Identifier or PSTD-ID. A PSTD-ID is a unique identifier for a data set. An example of a PSTD-ID for a directory entry is a Distinguished Name (DN). The SOAP driver uses the PSTD-ID for the association key. The Identifier Type defines the allowable PSTD-ID type. Select the Identifier Type to be used.

Table A-9 *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. In Designer, you must click the icon next to an option to edit it. This displays the Password Synchronization Options dialog that box that has a better display of the relationship between the different GCVs. In iManager, you should edit the Password Management Options on the Server Variables tab rather than under the GCVs. The Server Variables page has a better display of the relationship between the different GCVs. For more information about how to use the Password Management GCVs, see "Configuring Password Flow " in the *Identity Manager 3.6 Password Management Guide*.
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.

Using Java Extensions

B

The functionality of the SOAP driver 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:

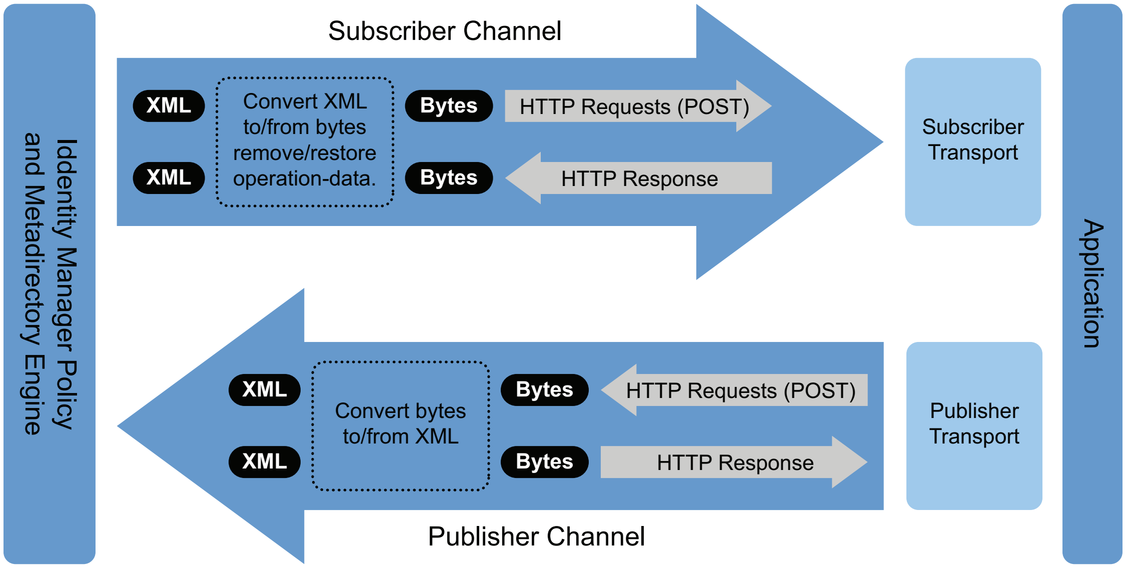
- Section B.1, "Overview," on page 49
- Section B.2, "Creating and Configuring Java Extensions," on page 50

B.1 Overview

If the application you are using with the SOAP driver 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 B-1, 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

Figure B-1 *Using Java to Extend Functionality*



The SOAP driver is 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/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.

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.

B.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 by 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 commands:

- **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 commands:

- **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 you can put any information here that you want to use during your class initialization.

9 Restart the driver.

You can now use your custom class.