NetIQ® Identity Manager

Driver for SOAP Implementation Guide

March 2018

Legal Notices

For information about NetIQ trademarks, see https://www.netiq.com/company/legal/.

Contents

About this Book and the Library

The *Identity Manager Driver for SOAP Implementation Guide* explains how to install and configure the Identity Manager Driver for SOAP.

Intended Audience

This book provides information for administrators implementing Identity Manager, application server developers, Web services administrators, and consultants, who also have an understanding of DSML/SPML, SOAP, and HTML.

Other Information in the Library

For more information about the library for Identity Manager, see the following resources:

- Identity Manager documentation website (https://www.netiq.com/documentation/identity-manager-47/)
- Identity Manager drivers documentation website (https://www.netiq.com/documentation/identity-manager-47-drivers/)

About NetIQ Corporation

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

Our Viewpoint

Adapting to change and managing complexity and risk are nothing new

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

Enabling critical business services, better and faster

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

Our Philosophy

Selling intelligent solutions, not just software

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

Driving your success is our passion

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

Our Solutions

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

Contacting Sales Support

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

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

Contacting Technical Support

For specific product issues, contact our Technical Support team.

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

Contacting Documentation Support

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

Contacting the Online User Community

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

1 Understanding the SOAP Driver

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:

Driver Concepts

This section contains the following information:

Data Management

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

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 the SPML 1.0, SPML 2.0, and DSML 2.0 protocols.

- **SPML:** 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 or SPML 2.0 depending on the sample configuration selected. SPML binds with SOAP 1.1 and uses HTTP and HTTPS 1.1 as the transport.
- **DSML:** 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 “Understanding the DSML Configuration” on page 29 and “Understanding the SPML Configuration” on page 29.

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.

HTTPS

HTTPS is the HTTP protocol over Secure Socket Layer (SSL) as a sub-layer under the regular HTTP application layering. HTTPS encrypts and decrypts user page requests as well as the pages that are returned by the Web server.

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.

Understanding Driver 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 `<driver-operation-data>`. The `<driver-operation-data>` element has two purposes. First, it can be used to match commands with the responses they generate, and can be used to create associations in transformation policies. Second, it can be used to override default Subscriber channel connection attributes.

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

By default, when the `<driver-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 `<driver-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.

NOTE: Earlier versions of the driver shim uses the `<operation-data>` element to achieve this purpose. The driver shim still supports this for backward compatibility. However, use of the `<operation-data>` element for this purpose will be depreciated in future releases.

NetIQ recommends that you use the <driver-operation-data> element.

To see how the <driver-operation-data> element works, see “Managing Driver Operation Data” on page 31.

Support for Standard Driver Features

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

Local Platforms

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

For information about the operating systems supported for the Identity Manager server, see the NetIQ Identity Manager Technical Information website.

Remote Platforms

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

For information about the operating systems supported for the Identity Manager server, see the NetIQ Identity Manager Technical Information website.

Password Synchronization Support

The SOAP driver is capable of synchronizing passwords.

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 through customized Java code

  For more information, see Appendix B, “Using Java Extensions,” on page 59.

2 Installing the Driver Files

By default, the SOAP driver files are installed on the Identity Manager server at the same time as the Identity Manager engine. The installation program extends the Identity Vault's schema and installs both the driver shim and the driver packages. It does not create the driver in the Identity Vault (see Chapter 3, "Creating a New Driver Object," on page 17) or upgrade an existing driver's configuration (see Chapter 4, "Upgrading an Existing Driver," on page 25).

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 Identity Manager server, using the instructions in Considerations for Installing Identity Manager Components in the *NetIQ Identity Manager Setup Guide for Linux* or Planning Your Installation in the *NetIQ Identity Manager Setup Guide for Windows*.
- Install the Remote Loader (required to run the driver on a non-Identity Manager server) and the driver files on a non-Identity Manager server where you want to run the driver. See Installing Identity Manager in the *NetIQ Identity Manager Setup Guide for Linux*.

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 Identity Manager server or a non-Identity Manager server that meets the system requirements for running the Remote Loader service. For information about the operating systems supported for the Remote Loader, see the NetIQ Identity Manager Technical Information website (https://www.netiq.com/products/identity-manager/advanced/technical-information/).

3 Creating a New Driver Object

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 15), you can create the driver in the Identity Vault. You do so by installing the driver packages and then modifying the driver configuration to suit your environment.

The SOAP driver comes with packages for the SPML protocol and for the DSML protocol. For information about the two protocols, see “SPML and DSML” on page 10, “Understanding the DSML Configuration” on page 29, and “Understanding the SPML Configuration” on page 29.

The following sections provide instructions to create the driver:

Creating the Driver Object in Designer

You create the SOAP driver object by installing the driver packages and then modifying the configuration to suit your environment. After you create and configure the driver, you need to deploy it to the Identity Vault and start it.

NOTE: You should not create driver objects by using the new Identity Manager 4.0 and later configuration files through iManager. This method of creating driver objects is no longer supported. To create drivers, you now need to use the new package management features provided in Designer.

Importing the Current Driver Packages

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

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

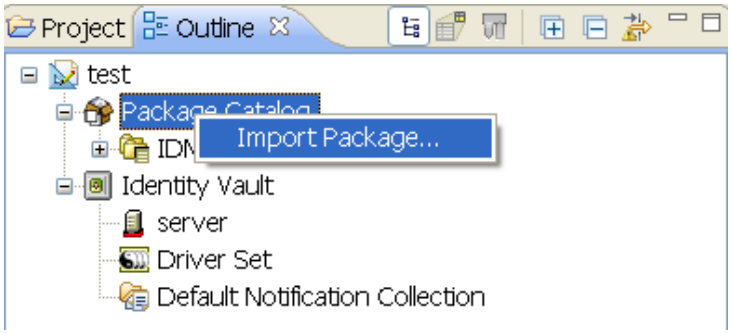
1. Open Designer.
2. In the toolbar, click **Help** > **Check for Package Updates**.
3. Click **OK** to update the packages

   or

Click **OK** if the packages are up-to-date.

4 In the Outline view, right-click the Package Catalog.

5 Click **Import Package**.



6 Select any SOAP driver packages

or

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

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

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

8 After the current packages are imported, continue with "Installing the Driver Packages" on page 18.

Installing the Driver Packages

After you have imported the current driver packages into the Package Catalog, you can install the driver packages to create a new driver.

1 In Designer, open your project.

2 In the Modeler, right-click the driver set where you want to create the driver, then click **New** > **Driver**.

3 Select **SOAP Base**, then click **Next**.

4 Select the type of SOAP driver packages to install, then click **Next**.

The options are:

- DSML 2.0
- SPML 1.0
- SPML 2.0
- Other

5 Select the optional features to install for the SOAP driver, then click **Next**.

The options are:

SOAP Password Synchronization: This packages contains the policies that enable the SOAP driver to synchronize passwords. If you want to synchronize passwords, verify that this option is selected. For more information, see the *NetIQ Identity Manager Password Management Guide*.

Managed System Information: This package contains the policies that enable Identity Reporting. For more information, see the *Administrator Guide to NetIQ Identity Reporting*.

6 (Conditional) If there are package dependencies for the packages you selected to install, you must install them to install the selected package. Click **OK** to install the package dependency listed.

7 (Conditional) If more than one type of package dependency must be installed, you are presented with these packages separately. Continue to click **OK** to install any additional package dependencies.

8 On the Driver Information page, specify a name for the driver, then click **Next**.

9 On the Install SOAP Base page, fill in the following fields for the Subscriber options, then click **Next**:

URL of the SOAP server or Web Service: Specify the URL of the SOAP server or Web service.

Authentication ID: Specify the ID used to authenticate to the SOAP server or Web service.

Authentication Password: Specify the password for the authentication ID.

Truststore file: Specify the path and the name of the keystore file that contains the trusted certificates for the remote server to provide server authentication. For example, `c:\security\truststore`. Leave this field blank when server authentication is not used.

Set mutual authentication parameters: Select **Show** if you want to set mutual authentication information.

- **Keystore file:** Specify the path and the name of the keystore file that contains the trusted certificates for the remote server to provide mutual authentication. For example, `C:\security\keystore`. Leave this field blank when mutual authentication is not used.
- **Keystore password:** Specify the password for the keystore file. Leave this field blank when mutual authentication is not used.

Proxy host and port: Specify the host address and the host post when a proxy host and port are used. For example: 192.10.1.3:18180. Choose an unused port number on your server. Otherwise, leave this field blank.

10 On the Install SOAP Base page, fill in the following fields for the Publisher options, then click **Next**:

Listening IP address and port: Specify the IP address of the server where this driver is installed and the port 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 incoming requests, processes the requests, and returns a result.

Authentication ID: Specify the authentication ID to validate incoming requests if Basic Authorization (on the HTTP header) is used.

Authentication Password: Specify the password for the authentication ID.

KMO name: When this server is configured to accept HTTPS connections, this is the KMO name in eDirectory. The KMO name is the name before the - in the RDN. Leave this field blank when a keystore file is issued or when HTTPS connections are not used.

Keystore file: When this server is configured to accept HTTPS connections, this is the path and the name of the keystore file. For example; `C:\security\keystore`. Leave this field blank when a KMO name is used or when HTTPS connections are not used.

Keystore password: When this server is configured to accept HTTPS connections, this is the keystore file password. Leave this field blank when a KMO name is used or when HTTPS connections are not used.

Server key alias: When this server is configured to accept HTTPS connections, this is the key alias. Leave this field blank 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 blank 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, select **Required**.

Heartbeat interval in minutes: Specify the heartbeat interval in minutes. Leave this field blank to turn off the heartbeat.

11 Fill in the following fields for the Remote Loader information, then click **Next**:

Connect To Remote Loader: Select **Yes** or **No** to determine if the driver will use the Remote Loader. For more information, see Configuring the Remote Loader and Drivers in the *NetIQ Identity Manager Setup Guide for Linux*.

If you select **No**, skip to Step 12. If you select **Yes**, use the following information to complete the configuration of the Remote Loader:

Host Name: Specify the IP address or DNS name of the server where the Remote Loader is installed and running.

Port: Specify the port number for this driver. Each driver connects to the Remote Loader on a separate port. The default value is 8090.

Remote Loader Password: Specify a password 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.

Driver Password: Specify a password for the driver to authenticate to the Identity Manager server. It must be the same password that is specified as the Driver Object Password on the Remote Loader.

12 (Conditional) This page is displayed only if you selected to install the Managed Systems Information packages. On the Install SOAP Managed System Information page, fill in the following fields to define your SOAP system, then click **Next**:

Name: Specify a descriptive name for this SOAP system. The name is displayed in reports.

Description: Specify a brief description for this SOAP system. The description is displayed in reports.

Location: Specify the physical location of this SOAP system. The location is displayed in reports.

Vendor: Leave Microsoft as the vendor of SOAP. This information is displayed in reports.

Version: Specify the version of this SOAP system. The version is displayed in the reports.

13 (Conditional) This page is displayed only if you selected to install the Managed Systems packages. On the Install SOAP Managed System Information page, fill in the following fields to define the ownership of the SOAP system, then click **Next**:

Business Owner: Select a user object in the Identity Vault that is the business owner of the SOAP system. This can only be a user object, not a role, group, or container.

Application Owner: Select a user object in the Identity Vault that is the application owner of the SOAP system. This can only be a user object, not a role, group, or container.

14 (Conditional) This page is displayed only if you selected to install the Managed System packages. On the Install SOAP Managed System Information page, fill in the following fields to define the classification of the SOAP system, then click **Next**:

Classification: Select the classification of the SOAP system. This information is displayed in the reports. You options are:

- Mission-Critical

- Vital
- Not-Critical
- Other

  If you select **Other**, you must specify a custom classification for the SOAP system.

Environment: Select the type of environment the SOAP system provides. The options are:

- Development
- Test
- Staging
- Production
- Other

  If you select **Other**, you must specify a custom classification for the SOAP system.

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

16 After you have installed the driver, you must change the configuration for your environment. Proceed to "Configuring the Driver Object" on page 21.

Configuring the Driver Object

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

- **Configure the driver parameters:** There are many settings that can help you customize and optimize the driver. The settings are divided into categories such as Driver Configuration, Engine Control Values, and Global Configuration Values (GCVs). Although it is important for you to understand all of the settings, your first priority should be to review the Driver Parameters located on the Driver Configuration page. The Driver Parameters let you configure the publication method and other parameters associated with the Publisher channel.
- **Customize the driver policies and filter:** The driver policies and filter control data flow between the Identity Vault and the application. You should ensure that the policies and filters reflect your business needs. For instructions, see Chapter 5, "Customizing the Driver," on page 29.
- **Set Up a Secure HTTPS Connection:** The connection between the driver and the SPML or DSML server can be configured to use a secure HTTPS connection rather than an HTTP connection. For instructions, see Chapter 6, "Securing Communication," on page 37.

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

Deploying the Driver Object

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

1 In Designer, open your project.

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

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

Host: Specify the IP address or DNS name of the server hosting the Identity Vault.

Username: Specify the DN of the user object used to authenticate to the Identity Vault.

Password: Specify the user's password.

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

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

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

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

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

6b Click **OK** twice.

For more information about defining a Security Equivalent User in objects for drivers in the Identity Vault, see “Establishing a Security Equivalent User” in the Identity Manager 4.0.2 Security Guide.

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

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

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

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

8 Click **OK**.

9 Continue with the next section, Starting the Driver.

Starting the Driver

When a driver is created, it is stopped by default. To make the driver work, you must start the driver. Identity Manager is an event-driven system, so after the driver is started, it won’t do anything until an event occurs. You can use iManager or dxevent commands to start the driver.

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**.

Activating the Driver

The Identity Manager driver for SOAP is part of the Identity Manager Integration Module for Tools. This integration module includes the following drivers:

- Identity Manager driver for Delimited Text
- Identity Manager driver for REST
- Identity Manager driver for SOAP

This integration module requires a separate activation. After purchasing the integration module, you will receive activation details in your NetIQ Customer Center.

If you create a new SOAP driver in a driver set that already includes an activated driver from this integration module, the new driver inherits the activation from the driver set.

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

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

Adding Packages to an Existing Driver

You can add new functionality to an existing driver by adding new packages to it.

1. Right-click the driver, then click **Properties**.
2. Click **Packages**, then upgrade the already installed SOAP Base package.
   - **2a** Select the package from the list of packages, then click the **Select Operation** cell.
   - **2b** Click **Upgrade** from the drop-down list, then click **Apply**.
   - **2c** Click **OK** to close the Package Management page.

     You can upgrade the Password Synchronization package in a similar way.
3. Click the **Add Packages** icon .
4. Select the packages to install.
5. (Optional) If you want to see all available packages for the driver, clear the **Show only applicable package versions** option, if you want to see all available packages for the driver, then click **OK**.

   This option is only displayed on drivers. By default, only the packages that can be installed on the selected driver are displayed.
6. Click **Apply** to install all of the packages listed with the Install operation.
7. (Conditional) Fill in the fields with appropriate information to install the package you selected for the driver, then click **Next**.
8. Read the summary of the installation, then click **Finish**.
9. Click **OK** to close the Package Management page after you have reviewed the installed packages.
10. Modify the driver configuration settings. See “Configuring the Driver Object” on page 21.
11. Deploy the driver. See “Deploying the Driver Object” on page 21.
12. Start the driver. See “Starting the Driver” on page 22.
13. Repeat Step 1 through Step 9 for each driver where you want to add the new packages.

4 Upgrading an Existing Driver

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

What’s New in Version 4.1.0

This version of the driver does not provide any new features.

What’s New in Version 4.0.0.4

Version 4.0.0.4 of the driver provides support for configuring Suite B communication between the Remote Loader and the Identity Manager engine. For more information, see “Authentication” on page 52. For more information about Suite B, see Suite B Cryptography.

Upgrade Procedure

The driver upgrade process involves upgrading the installed driver packages and updating the existing driver files. These are independent tasks and can be separately planned for a driver. For example, you can update the driver packages and choose not to update the driver files at the same time. However, you are recommended to complete all the update steps within a short amount of time to ensure that the driver has the latest updates.

Before starting the upgrade process, ensure that you have taken a back-up of the current driver configuration.

Upgrading the Installed Packages

1. Download the latest available packages.

   To configure Designer to automatically read the package updates when a new version of a package is available, click **Windows > Preferences > NetIQ > Package Manager > Online Updates** in Designer. However, if you need to add a custom package to the Package Catalog, you can import the package `.jar` file. For more information about creating custom packages, see Upgrading Installed Packages in *NetIQ Designer for Identity Manager Administration Guide*.

2. Upgrade the installed packages.

   2a Open the project containing the driver.

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

2c Click **Packages**.

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

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

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

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

NOTE: Designer lists all versions available for upgrade.

2g Click **Apply**.

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

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

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

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

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

Applying the Driver Patch

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

Prerequisites

Before installing the patch, complete the following steps:

1. Take a back-up of the current driver configuration.
2. (Conditional) If the driver is running with the Identity Manager engine, stop the Identity Vault and the driver instance.
3. (Conditional) If the driver is running with a Remote Loader instance, stop the Remote Loader instance and the driver instance.
4. In a browser, navigate to the NetIQ Patch Finder Download Page.
5. Under **Patches**, click **Search Patches**.
6. Specify **Identity Manager *nn* SOAP Driver nn** in the search box.
7. Download and unzip the contents of the patch file to a temporary location on your server.

   For example, `IDM45_SOAP_4004.zip`.

Applying the Patch as a Root User

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

1. Update the driver files:
   - **Linux:** Log in to your server as `root` and run the following command in a command prompt:

     `rpm -Uvh <Driver Patch File Temporary Location>/linux/novell-DXMLsoap.rpm`

     For example, `rpm -Uvh <IDM45_SOAP_4004.zip>/linux/novell-DXMLsoap.rpm`

- **Windows:** Navigate to the <*Extracted Driver Patch File Temporary Location*>\windows folder and copy the following files to <IdentityManager installation>\NDS\lib or <IdentityManager installation>\RemoteLoader\<architecture>\lib folder.
  - SOAPShim.jar
  - SOAPUtil.jar

2 (Conditional) If the driver is running locally, start the Identity Vault and the driver instance.

3 (Conditional) If the driver is running with a Remote Loader instance, start the Remote Loader instance and the driver instance.

Applying the Patch as a Non-Root User

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

The _db.000 file is created during a non-root installation of the Identity Manager engine. Absence of this file might indicate that Identity Manager is not properly installed. Reinstall Identity Manager to correctly place the file in the directory.

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

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

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

3 Download the patch and untar or unzip the downloaded file.

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

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

For example, to install the SOAP driver RPM, use this command:

```
rpm --dbpath $ROOTDIR/rpm -Uvh --relocate=/usr=$ROOTDIR/opt/novell/eDirectory
--relocate=/etc=$ROOTDIR/etc --relocate=/opt/novell/eDirectory=$ROOTDIR/opt/
novell/eDirectory --relocate=/opt/novell/dirxml=$ROOTDIR/opt/novell/dirxml --
relocate=/var=$ROOTDIR/var --badreloc --nodeps --replacefiles /home/user/
novell-DXMLsoap.rpm
```

5 Customizing the Driver

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

Understanding the DSML Configuration

The DSML package 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 that are delivered in the DSML package.

The DSML package 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.

  This sample has 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 is 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.

Understanding the SPML Configuration

The SPML package 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 that are delivered in the SPML package.

The SPML package does the following:

- Provides generic SPML functionality based on the *OASIS SPML V2 core xsd* standard.

The SPML package does not pair with any specific SPML application. In some specific implementations, you can modify the default policies to conform with the provider standards. For example, the `containerID` attribute is optional as per the *OASIS SPML V2 core xsd* standard, whereas it is mandatory for *Quest One ActiveRoles SPML Provider*.

- 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 is 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.

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 Identity Manager 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.

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 in the SOAP packages. For more information on style sheets see "Defining Policies by Using XSLT Style Sheets" in the *NetIQ Identity Manager Understanding Policies Guide*.

Managing Driver Operation Data

The driver shim applies special handling to Subscriber commands based on the `<driver-operation-data>` element. On the Subscriber channel, the `<driver-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 "Understanding Driver Operation Data" on page 11, the `<driver-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 `<driver-operation-data>` element (and all child elements) to the resulting response.

```
<nds dtdversion="4.0" ndsversion="8.x">
   <source>
      <product edition="Advanced" version="4.0.2.0">DirXML</product>
         <contact>Novell, Inc.</contact>
    </source>
           <input>
            <user email="thomas.wagner@example.COM" enabled="true"
ns1:password="Password1!" username="thomas.wagner"
xmlns:ns1="http://docs.openstack.org/identity/api/ext/OS-KSADM/v1.0"
xmlns:ns2="http://docs.openstack.org/identity/api/v2.0">
           <driver-operation-data>
                 <request-headers remove-existing="true">
                    <request-header name="accept">application/xml
                    </request-header>
                 <request-header
             name="X-Auth-Token">262f9eefc1e9488fa65474c5aa0f5ca1</request-header>
                 </request-headers>
           </driver-operation-data>
          </user>
         </input>
</nds>
```

`<driver-operation-data>` must be a child element of the operation node which is the `<user>` node in this example. Otherwise, `<user>` and `<driver-operation-data>` are considered as separate operations and will not be applied to the `<user>` operation.

Consider another example below.

```
<input>
    <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
        <soapenv:Header>
            <soapenv:Body>
                <ser:findRoleByExampleWithOperatorRequest xmlns:ser="http://
www.novell.com/role/service">
                    <ser:role>
                        <ser:name>abRole</ser:name>
                    </ser:role>
                    <ser:operator>false</ser:operator>
                </ser:findRoleByExampleWithOperatorRequest>
            </soapenv:Body>
        </soapenv:Header>
        <driver-operation-data VALUE="true" VALUE2="This is valid. driver-
operation-data can also be in the child nodes also from here." />
    </soapenv:Envelope>
    <driver-operation-data VALUE="true" VALUE2="This should not be placed here. It
does not satisfy condition 1" />
</input>
```

In this example, `<driver-operation-data>` is a child element of `<soapenv:Envelope>` operation node.

- “Using Driver Operation Data to Specify XML to Be Returned on the Result” on page 32
- “Using Driver Operation Data to Override Default Subscriber Options” on page 33

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

The `<driver-operation-data>` is appended as a child element of the root node when it is restored on the response. You can override this by providing one or more parent-node-*n* attributes to the `<driver-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. If the attribute is found, the attribute is checked to see if the named node exists. If the node is found, the driver shim uses it as the parent for the `<driver-operation-data>` element on the response.

The following example shows the usage of parent-node-*n* in driver-operation-data when requesting and returning commands:

```
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
      <soap-env:Body>
        <batchRequest xmlns="urn:oasis:names:tc:DSML:2:0:core" xmlns:xsd="http://
www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
          <searchRequest derefAliases="neverDerefAliases"
dn="data\users\so1user120" requestID="0" scope="baseObject" sizeLimit="100">
            <filter>
              <and>
                <equalityMatch name="objectclass">
                  <value>inetOrgPerson</value>
                </equalityMatch>
              </and>
            </filter>
            <attributes>
              <attribute name="1.1"/>
            </attributes>
            <driver-operation-data parent-node-1="searchResponse" parent-node-
2="errorResponse">
              <return-to-me class-name="inetOrgPerson" command="query" dest-
dn="data\users\so1user120" event-id="0" scope="entry"/>
            </driver-operation-data>
          </searchRequest>
        </batchRequest>
      </soap-env:Body>
    </soap-env:Envelope>
```

The driver shim checks for `<searchResponse>` or `<errorResponse>` nodes in the applications' response. If one of these nodes is present in the response, the driver shim appends the `<driver-operation-data>` as a child element of the `<searchResponse>` node similar to below.

```
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
      <soap-env:Body>
        <batchResponse xmlns="urn:oasis:names:tc:DSML:2:0:core" xmlns:xsd="http://
www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
          <searchResponse>
            <searchResultDone requestID="searchRequest">
              <resultCode code="0" descr="success" xmlns=""/>
            </searchResultDone>
            <driver-operation-data parent-node-1="searchResponse" parent-node-
2="errorResponse">
              <return-to-me class-name="inetOrgPerson" command="query" dest-
dn="data\users\so1user120" event-id="0" scope="entry"/>
            </driver-operation-data>
          </searchResponse>
        </batchResponse>
      </soap-env:Body>
    </soap-env:Envelope>
```

Using Driver 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 34
- "Overriding Single Subscriber Options" on page 34
- "Overriding the Authorization Header" on page 35

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 by using the `<driver-operation-data>` element to specify which connection set to use for the current command.

To use the `<driver-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 if the numbers you use are incremental without gaps.
4. Add an attribute to the `<driver-operation-data>` element called `connection` and give it the value of the connection number you want to use.

   For example:

```
<driver-operation-data connection="2">
  ...(other driver-operation-data elements)
</driver-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 `<driver-operation-data>` element. The following table lists the attributes that can be used and the Subscriber option they are meant to override.

<driver-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:

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

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

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

Overriding the Authorization Header

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

Example:

```
<driver-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>
</driver-operation-data>
```

By default, the driver encodes the Authorization header value sent as part of the `<driver-operation data>`. In this scenario, you must specify the authorization information in clear text format.

Example:

```
<driver-operation-data>
    <request-headers remove-existing="true">
        <request-header name="Authorization">Basic
cn=admin,o=n:n</request-header>
        </request-headers>
</driver-operation-data>
```

If you do not want the driver to encode the Authorization header value, set the flag `"encode=false`". In this scenario, you must specify the authorization information in base64 format.

Example:

```
<driver-operation-data>
    <request-headers remove-existing="true">
        <request-header encode="false" name="Authorization">Basic
c2VydmVyMTpub3ZlbGw=</request-header>
    </request-headers>
</driver-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 the `<driver-operation-data>` or should replace the existing request-headers.

- If the remove-existing flag is set to true, the set of request-headers defined in the `<driver-operation-data>` replaces the existing ones defined in the Subscriber option.
- If the remove-existing flag is set to false, the set of request-headers defined in the Subscriber options is used in addition to the new headers defined in the `<driver-operation-data>`.

If the Authorization header already exists, it is overridden. Otherwise, it is added as new.

6 Securing Communication

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

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

The following sections provide instructions for creating a secure connection:

Configuring the Publisher Channel

1 Create a server certificate in iManager.:

- **1a** In the **Roles and Tasks** view, click **NetIQ 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**, then browse to a known location on your computer.
- **2i** Click **Save**, then click **Close**.

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

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

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

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

3b Enter the following command at a command prompt:

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

For example:

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

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

4a In 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**.

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 "Configuring the Publisher Channel" on page 37 for an example.

1 Make sure you have a server certificate signed by a certificate authority.

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

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

For example:

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

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

3 Configure the Subscriber channel to use the trust store you created in Step 2:

3a In iManager, in the **Roles and Tasks** view, click **Identity Manager > Identity Manager Overview**.

3b Locate the driver set containing the SOAP driver, then click the driver's icon to display the Identity Manager Driver Overview page.

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

3d In the **Keystore File** setting, specify the path to the trust store you created in Step 2.

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

NOTE: To use TLSv1 instead of SSLv3 in the HTTP client, change the JVM setting for the driver by using one of the following methods:

In **Designer**, right-click the **driver set** containing this driver. Click **Properties >Java** and set the **JVM option** as Dhttps.protocols=TLSv1 in the window that opens up. Click **Apply** and then click **OK**.

In **iManager**, navigate to the **Edit Driver Set properties** page, click the **Misc** tab and set the **JVM option** as `Dhttps.protocols=TLSv1`.

If the driver is using the Remote Loader, set the -javaparam option to DHOST_JVM_OPTIONS=-Dhttps.protocols=TLSv1 in the configuration file.

A driver with this setting will always initiate a connection only through the TLSv1 protocol and will not connect to servers using SSlv3 protocol.

7 Managing the Driver

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 *NetIQ Identity Manager Driver Administration Guide*.

8 Troubleshooting the Driver

You can log Identity Manager events by using the Event Auditing Service. 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 *Administrator Guide to NetIQ Identity Reporting*.

This section contains the following information on error messages:

Driver Shim Errors

The following errors 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: Messages in the 200-299 range indicate success.

Action: No action required.

Level: Success

Other HTTP Error Messages

Source: The status log or DSTrace screen.

Explanation: Other numerical error codes result in an error message containing the 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 “Subscriber Options” on page 54 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 Identity Manager engine calls the DriverShim.getSchema() method of the driver, and the driver is not using the SchemaReporter customization.

Action: A Java class needs to be written that implements the SchemaReporter interface, and the driver needs to be configured to load the class as a Java extension.

Level: Warning

Subscriber.execute() was called but the Subscriber was not configured correctly. The command was ignored.

Source: The status log or DSTrace screen.

Explanation: The Subscriber channel of the driver isn’t initialized properly. The driver continues to run but displays this message each time an event is received by the Subscriber channel.

Possible Cause: An improperly formatted driver configuration.

Action: Configure the driver correctly. See Chapter 5, “Customizing the Driver,” on page 29 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 “Publisher Options” on page 55 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 “Publisher Options” on page 55 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 “Publisher Options” on page 55 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

SOAP driver waits indefinitely on a response from the SOAP service

Source: The status log or DSTrace screen.

Explanation: The driver continues to send the information to the Web server but does not receive any response and appears to be in the waiting state. You can verify this state in the trace log file.

Possible Cause: SOAP service does not provide a response for the transaction when communicating with the SOAP driver.

Action: Perform one of the following actions:

- Restart eDirectory, restart the webservice, and then restart the driver.
- Pass the additional request-headers to the http post, which will result in the SOAP driver waiting on a response from the webservice for a finite number of seconds and then continuing to the next operation.

  Example: Use the following operation-data element to pass the additional request-header for the driver to wait for 60 seconds and then continue:

```
<operation-data>
<request-headers remove-existing="false">
<request-header name="Expect">60-continue</request-header>
</request-headers>
</operation-data>
```

Level: Fatal

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 section and you are using Java extensions, check with whomever provided you with the extensions for a list of error codes for that particular extension.

Level: Varies

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 *NetIQ Identity Manager Driver Administration Guide*.

9 Known Issues

The following known issues exist for this version of the driver:

No Response from the Connected System Hangs the Driver Shim

If the SOAP driver does not get a response to the command the driver sends to the connected system, the driver shim hangs.

Driver Needs Remote Loader IP Address in the Server Certificate Parameter in a Distributed or Clustered Environment

The SOAP driver fails to connect with the Remote Loader if the IP address of the Remote Loader is not specified in the default server certificate provided to the Remote Loader.

To workaround this issue, specify the IP address of the Remote Loader in the **Subject Alternative Names** parameter in iManager using the following steps:

1. Log in to iManager.
2. Click **NetIQ Certificate Server > Create Server Certificate**.
3. Create a custom server certificate and specify the Remote Loader's IP address in **Subject Alternative Names** in Step 4 of the certificate creation method.
4. Accept the rest of the certificate defaults.
5. Review the summary, click **Finish**, then click **Close**.

A Driver Properties

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 *NetIQ Identity Manager 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 a Designer icon.

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:

Driver Module

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

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:

- **Remote Loader Client Configuration for Documentation**: Includes information on the Remote Loader client configuration when Designer generates documentation for the driver.
- **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.

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.

Authentication

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

Authentication ID: This option is not used with the SOAP driver. The SOAP driver requires separate authentication settings for both the “Subscriber Options” on page 54 and the “Publisher Options” on page 55.

Authentication Context: This option is not used with the SOAP driver.

Remote Loader Connection Parameter: Used only if the driver is connecting to the application through the Remote Loader.

In iManager, enter `hostname=xxx.xxx.xxx.xxx port=xxxx secureprotocol=TLS version enforceSuiteB=`*`true/false`* `kmo=certificatename.`

- `hostname` specifies the IP address of the Remote Loader server.
- `port` specifies the TCP/IP port on which the Remote Loader listens for connections from the remote interface shim. The default port for the Remote Loader is 8090.
- `secureprotocol` specifies the version of the TLS protocol that the Remote Loader uses to connect to the Identity Manager engine. Identity Manager supports TLSv1, TLS v1_1, and TLSv1_2 versions only.
- `enforceSuiteB` specifies whether the Remote Loader uses Suite B for communicating with the Identity Manager engine. To use Suite B, specify `enforceSuiteB=true`. The communication supports only TLS version 1.2 version. Communication is not established if the connection has non-Suite B authentication algorithms.
- The `kmo` entry is optional. Use it only when an SSL connection exists between the Remote Loader and the Identity Manager engine.

  For example: `hostname=10.0.0.1 port=8090 kmo=IDMCertificate`

Application Password: This option is not used with the SOAP driver.

Remote Loader 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.

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.

Startup Option

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

Auto start: The driver starts every time the Identity Manager server is started.

Manual: The driver does not start when the Identity Manager server is started. The driver must be started through Designer or 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 applies only if the driver is deployed and was previously disabled. If this option is not selected, the driver re-synchronizes the next time it is started.

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:

- "Driver Options" on page 53
- "Subscriber Options" on page 54
- "Publisher Options" on page 55

Driver Options

<nds>, <input>, <output> Element Handling: Specify **Remove/add elements** if you want the driver shim to remove and add the required `<nds>`, `<input>`, and `<output>` XML elements.

query-ex operation supported: Select **Yes** if you want the driver shim to report support for the query-ex operation to the engine. Select **Yes** only if you are adding explicit support in your driver policy and transforms for the query-ex feature and if it can be supported by your target application or Web service. Most SOAP driver implementations should be set to **No**.

Custom Java Extensions: Select **Show** if you have developed custom Java classes to extend the driver shim's functionality. Otherwise, select **Hide**.

- **Document Handling:** Select **Implemented** if you have developed a custom Java class to process data as XML documents.
- **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.

For more information, see Appendix B, “Using Java Extensions,” on page 59.

Subscriber Options

URL of the SOAP server or Web Service: 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.

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:. 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.

- **Keystore file:** Specify the path and the name of the keystore file that contains the trusted certificates for the remote server to provide mutual authentication. For example, `C:\security\keystore`. Leave this field blank when mutual authentication is not used.
- **Keystore password:** Specify the password for the keystore file. Leave this field blank when mutual authentication is not used.

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.

HTTP errors to retry: Specify the HTTP errors that must return a retry status. Error codes must be a list of integers separated by spaces. For example, 307 404 408 503 504.

Response Time out: Specify the time in milliseconds for getting a response from the remote SOAP server after which the request returns an error.

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 **Use** 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 Options

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 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 or when HTTPS connections are not used.

Keystore password: Specify the keystore file password used with the Keystore file: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.

NOTE: A SOAP client calling the Web service in the Publisher channel must specify a URL ending with a slash. For example, `http://1.1.1.1:9095/`. Without a context path (the slash), the driver does not process the request received.

ECMAScript

Displays an ordered list of ECMAScript resource files. The files contain extension functions for the driver that Identity Manager loads when the driver starts. You can add additional files, remove existing files, or change the order the files are executed.

Global Configuration

Displays an ordered list of Global Configuration objects. The objects contain extension GCV definitions for the driver that Identity Manager loads when the driver is started. You can add or remove the Global Configuration objects, and you can change the order in which the objects are executed.

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-clickthe driver set icon  , then click **Properties > GCVs**.

The global configuration values are organized as follows:

- “Password Synchronization” on page 57

Password Synchronization

These GCVs enable password synchronization between the Identity Vault and the connected system.

In Designer, you must click the  icon next to a GCV to edit it. This displays the Password Synchronization Options dialog box for a better view of the relationship between the different GCVs.

In iManager, to edit the Password management options go to **Driver Properties > Global Configuration Values**, and then edit it in your Password synchronization policy tab.

For more information about how to use the Password Management GCVs, see “Configuring Password Flow” in the *NetIQ Identity Manager Password Management Guide*.

Connected System or Driver Name: Specify the name of the SOAP system or the driver name. This valued is used by the e-mail notification template to identity the source of the notification message.

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, attempts to reset the password in the connected system by using the Distribution Password from the Identity Manager data store.

Notify the user of password synchronization failure via e-mail: If **True**, notifies the user by e-mail of any password synchronization failures.

B Using Java Extensions

The functionality of the SOAP driver can be extended by using Java. You use an API defined by Java interfaces, to 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.

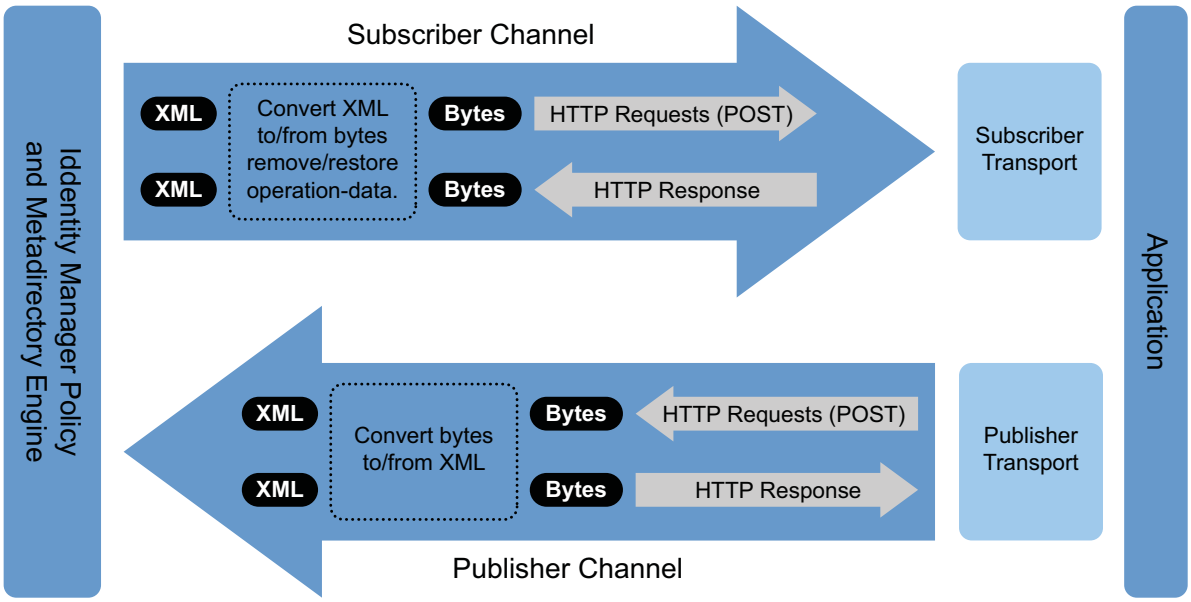
This section contains the following information on using Java extensions:

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 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.

Creating and Configuring Java Extensions

Using the sample code and SOAP Driver Javadoc found at the NetIQ Developer Downloads Web site 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 for 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 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.

C Trace Levels

The driver supports the following trace levels:

Table C-1 *Supported Trace Levels*

Level	Description
0	No debugging
1-3	Identity Manager messages. Higher trace levels provide more detail.
4	Previous level plus Remote Loader, driver, driver shim, and driver connection messages, driver parameters, driver security, driver schema, request and response XML

For information about setting driver trace levels, see "Viewing Identity Manager Processes" in the *NetIQ Identity Manager Driver Administration Guide*.