Novell Nsure™ Identity Manager Driver for SOAP

1.0

www.novell.com

IMPLEMENTATION GUIDE

May 18, 2005

Novell®

Legal Notices

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

www.novell.com

Nsure Identity Manager Driver for SOAP Implementation Guide
May 18, 2005

Online Documentation: To access the online documentation for this and other Novell products, and to get updates, see www.novell.com/documentation.

Novell Trademarks

DirXML is a registered trademark of Novell, Inc. in the United States and other countries.
NetWare a registered trademark of Novell, Inc. in the United States and other countries.
Novell is a registered trademark of Novell, Inc. in the United States and other countries.
Nsure is a registered trademark of Novell, Inc. in the United States and other countries.
SUSE is a registered trademark of SUSE AG, a Novell business.

Third-Party Materials

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

Contents

About This Guide 7

1 Overview 9

Driver Concepts . . . 9
- Data Management . . . 9
- How the Driver Works . . . 10
- Understanding Operation Data . . . 11

Driver Features . . . 12

2 Installing the Driver 13

Driver Prerequisites . . . 13
Installing the Driver . . . 13
- Installing on Windows . . . 14
- Installing on Linux, Solaris, or AIX . . . 14

3 Using the Sample Driver Configurations 15

Creating a Driver Object Using a Driver Configuration File . . . 15
Understanding the SPML Configuration . . . 17
Understanding the DSML Configuration . . . 18

4 Configuring the Driver 19

Configuring Driver Settings . . . 19
Configuring Subscriber Settings . . . 22
- Configuring the Subscriber to Make HTTPS Connections to the Remote Web Service . . . 23
- Configuring the Subscriber to Use a Proxy . . . 23

Configuring Publisher Settings . . . 24
- Configuring the Publisher to Receive HTTPS Connections . . . 24

Creating XSLT Style Sheets . . . 25
- Using <operation-data> with Style Sheets . . . 25

5 Using the Driver 27

Starting the Driver . . . 27
Migrating and Resynchronizing Data . . . 27
Activating the Driver . . . 28

6 Troubleshooting the Driver 29

Driver Shim Errors . . . 29
Java Customization Errors . . . 31

A Using Java Extensions 33

Overview . . . 33
Creating and Configuring Java Extensions . . . 34

About This Guide

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

- “Overview” on page 9
- “Installing the Driver” on page 13
- “Using the Sample Driver Configurations” on page 15
- “Configuring the Driver” on page 19
- “Troubleshooting the Driver” on page 29

Documentation Conventions

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

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

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

User Comments

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

Additional Documentation

For the most recent version of this document, see Nsure Identity Manager Driver 1.0 for SOAP in the Driver Implementation Guides (http://www.novell.com/documentation/dirxmldrivers/index.html) section.

Documentation Updates

For information on Nsure™ Identity Manager, see the Identity Manager Documentation Web site (http://www.novell.com/documentation/dirxml20/index.html).

For information on other Identity Manager drivers, see Driver Implementation Guides (http://www.novell.com/documentation/dirxmldrivers/index.html).

1 Overview

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 Nsure Identity Manager Driver 1.0 for SOAP:

- "Driver Concepts" on page 9
- "Driver Features" on page 12

Driver Concepts

This section contains the following information:

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

Data Management

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

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

SOAP

SOAP (Simple Object Access Protocol) is an XML-based protocol for exchanging messages located in Identity Manager. 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 uses two protocols to handle requests and responses: SPML 1.0 and DSML 2.0. These are the service-oriented protocols located in Identity Manager.

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

  The driver supports SPML 1.0. SPML binds with SOAP 1.1 and uses HTTP and HTTPS 1.1 as the transport.

- **DSML 2.0:** Directory Services Markup Language represents directory structural information, directory queries and updates, and the results of these operations as XML documents.

DSML binds with SOAP 1.1 and uses HTTP(S) 1.1 as the transport.

For more information about the sample SPML and DSML configurations included with the driver, see "Using the Sample Driver Configurations" on page 15.

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.

How the Driver Works

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

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 Operation Data

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

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

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

To see how Operation Data works with the style sheets, see "Using <operation-data> with Style Sheets" on page 25.

Driver Features

The driver contains the following features:

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

  By default, a basic authorization request header with an ID and password is provided for the Subscriber channel. For more information, see “Creating a Driver Object Using a Driver Configuration File” on page 15.
- SSL connections using the HTTPS protocol
- Subscriber HTTP and HTTPS proxy servers
- Definition and slection of multiple subscriber connections in the policy at run time
- Potential to act as an HTTP or HTTPS listener for incoming connections on the publisher channel
- Potential extensibility using customized Java code

  For more information, see Appendix A, “Using Java Extensions,” on page 33.

2 Installing the Driver

The section contains the following information about installing the driver:

- "Driver Prerequisites" on page 13
- "Installing the Driver" on page 13

Driver Prerequisites

- ❑ One of the following operating systems:
  - NetWare® 6 or 6.5 with the latest Support Pack
  - Windows* NT*, 2000, or 2003 with the latest Service Patch
  - Linux Red Hat* AS, ES 2.1, or AS 3.0
  - SUSE® LINUX Enterprise Server 8 or 9 (including SP1)
  - Solaris* 8 or 9
  - AIX 5.2L
- ❑ Novell® eDirectory™ 8.7.3 with the latest Support Pack
- ❑ Novell Nsure™ Identity Manager 2.0.1 or higher
- ❑ Novell iManager 2.0.2 or higher

Installing the Driver

- "Installing on Windows" on page 14
- "Installing on Linux, Solaris, or AIX" on page 14

Installing on Windows

1. Locate the SOAP driver download at http://download.novell.com (http://download.novell.com/index.jsp).

   Use the Keyword search to locate the driver (Identity Manager Integration Module for Tools.)
2. Follow the on-screen download instructions.
3. Double-click soapinstall.exe.
4. Follow the installation wizard to install the components on your systems.
5. Click Finish to close the installation program.

After installation you must set up the driver as explained in "Creating a Driver Object Using a Driver Configuration File" on page 15.

Installing on Linux, Solaris, or AIX

1. To launch the installation program for your platform, type one of the following commands at the command prompt:

Platform	Command
Linux	/soapinstall_linux.bin **TIP:** /soapinstall_linux.bin -i console launches the installation program in text mode.
Solaris	/soapinstall_solaris.bin
AIX	/soapinstall_aix.bin

2. Follow the installation wizard to install the components on your systems.

   To return to a previous page, type **`previous`**, then press Enter.

   To continue, press Enter.
3. Click Finish to close the installation program.

After installation, you must set up the driver as explained in "Creating a Driver Object Using a Driver Configuration File" on page 15.

3 Using the Sample Driver Configurations

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

This section covers the following topics:

- "Creating a Driver Object Using a Driver Configuration File" on page 15
- "Understanding the SPML Configuration" on page 17
- "Understanding the DSML Configuration" on page 18

Creating a Driver Object Using a Driver Configuration File

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

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

For more information about the sample files, see "Understanding the SPML Configuration" on page 17 and "Understanding the DSML Configuration" on page 18.

To create a Driver object using a driver configuration file:

1. Identify a user object that has the rights that the driver needs to have on the server.

   The tendency is to use the Admin user object for this task. However, you might want to create a DriversUser (for example) and assign security equivalence to that user. Whatever rights that the driver needs to have on the server, the DriversUser object must have the same security rights.

   If necessary, a new DriversUser object should be created in iManager.

2. Create a new driver object or import the configuration onto an existing driver.

   In Novell iManager, select DirXML Utilities, then use one of the tasks as described in "Managing DirXML Drivers" (http://www.novell.com/documentation/dirxml20/index.html?page=/documentation/dirxml20/admin/data/anfke45.html).

   The wizard prompts you to provide the following information:

   NOTE: You need to scroll down to see these fields.

Field	Description
Driver Name	Specify the name of the driver object in Identity Manager.

Field	Description
Configure Data Flow	Specify the driver channels you want to be active. **eDirectory to DSML:** Sends Identity Vault events to the application. **DSML to eDirectory:** Receives events from the application. **Bi-Directional:** Activates both the eDirectory™ and the DSML channels.
<nds>, <input>, <output> Element Handling	Select one of the following: **Remove/Add Elements:** The driver shim removes and adds the required XML elements of nds, input, and output. These required elements are removed from XML documents sent to the application and are added to XML documents received from the application before sending the document to the metadirectory (Identity Manager) engine. This is the preferred option for the SOAP Driver. **Pass Elements Through:** Turns off element handling. The required XML elements of nds, input, and output aren't added or removed to XML documents as necessary.
Driver is Local/Remote	Select one of the following: **Local:** Runs the driver shim from the server holding the driver set. **Remote:** Runs the driver from a remote server using the Remote Loader. If you specify this option, click Next, then specify Remote Loader configuration information. For more information, see "Setting Up Remote Loaders" (http://www.novell.com/documentation/dirxml20/index.html?page=/documentation/dirxml20/admin/data/bs35pip.html).
(Conditional) Subscriber Channel fields **NOTE:** These fields are displayed only if you selected Subscriber Channel Only or Both Channels in the Driver Channels to Activate field.	**URL of the Remote Server:** Specify the URL of the remote server and the port number that the server listens on, for example, http://137.66.10.13:18180/soap. The server is a software component that listens for, processes, and returns the results for valid requests. **TIP:** If you configure the driver to use SSL, the URL must begin with https rather than http. **(Conditional) Authentication ID:** If the remote server requires an authentication ID, specify it in the field. Otherwise leave the field empty. **(Conditional) Authentication Password and Re-enter the Password:** Specify the authentication password for the remote server if you specified an Authentication ID above. Otherwise, leave these fields empty.

Field	Description
(Conditional) Publisher Channel fields **NOTE:** These fields are displayed only if you selected Publisher Channel Only or Both Channels in the Driver Channels to Activate field.	**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. You can specify 127.0.0.1 if there is only one network card installed in the server. Choose an unused port number on your server, for example, 127.0.0.1:18180. The driver listens on this address for requests, processes the requests, and returns a result. **(Conditional) Authentication ID:** Specify the authentication ID of the remote server to validate incoming requests. If the remote server does not send an Authentication ID, leave this field empty. **(Conditional) Authentication Password and Re-enter the Password:** Specify the authentication password of the remote server to validate incoming requests if you specified an Authentication ID above. Otherwise, leave these fields empty.

3. Define security equivalences using the user object identified in Step 1 on page 15.
4. Identify all objects that represent administrative roles and exclude them from replication.

   Exclude the security-equivalence object (for example, DriversUser) that you specified in Step 2. If you delete the security-equivalence object, you have removed the rights from the driver. Therefore, the driver can't make changes to Identity Manager.
5. Click Finish.
6. Configure additional settings for the driver.

   For more information, see “Configuring the Driver” on page 19.

Understanding the SPML Configuration

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

The sample SPML import file does the following:

- Provides generic SPML functionality.

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

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

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

  The configuration handles all queries as SPML scope = “subtree” and uses the entry and subordinate scope concepts. Specific query operations can be handled through policy and style sheet customization.

- Supports string, structured and Distinguished Name (DN) attribute types.
- Supports password set operation.

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

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

Understanding the DSML Configuration

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

The sample DSML import file does the following:

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

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

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

  Specific query operations can be handled through policy and style sheet customizations.
- Supports password set operation.

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

4 Configuring the Driver

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

- "Configuring Driver Settings" on page 19
- "Configuring Subscriber Settings" on page 22
- "Configuring Publisher Settings" on page 24
- "Creating XSLT Style Sheets" on page 25

Configuring Driver Settings

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

     com.novell.nds.dirxml.driver.soap.SOAPDriver
5. Specify driver object password information:
   - **5a** Scroll to the Driver Object Password section.
   - **5b** In the fields, enter and re-enter the driver object password.
6. Specify authentication information:
   - **6a** Scroll to the Authentication section.
   - **6b** Specify the requested Authentication information.
7. Specify startup information:
   - **7a** Scroll to the Startup section.
   - **7b** Select one of the following:
     - **Auto Start:** The driver starts automatically when eDirectory starts.
     - **Manual:** The driver must be started manually using iManager.
     - **Disabled:** The driver will not run.
8. Specify the following driver settings:

Section	Field	Description
Driver Settings	<nds>, <input>, <output> Element Handling	Specify [Remove/add elements] if you want the driver shim to remove and add the required XML elements <nds>, <input>, and <output>. The required elements are removed from XML documents sent to the application and are added to XML documents received from the application before presenting the document to the metadirectory engine. Otherwise, specify [Pass elements through] to turn off this element handling.
	Custom Java Extensions	Specify Show if you have developed custom Java* classes to extend the driver shim's functionality. Otherwise, specify Hide. For more information, see Appendix A, “Using Java Extensions,” on page 33.
Subscriber Settings	URL of the Remote DSML Server	Enter the URL of the remote server and the port number that the server listens on. The URL should begin with *http://* unless you have configured SSL settings, in which case it should begin with *https://* and use a DNS hostname rather than an IP address.
	(Conditional) Authentication ID	If the remote server requires an authentication ID, enter it in the field. Otherwise, leave the field empty.
	(Conditional) Authentication Password and Re-Enter Authentication Password	Enter the authentication password for the remote server if you entered an Authentication ID above. Otherwise, leave the field empty.
	Remove Existing Password	Click the box to remove the existing password. Then specify the new password in the Authentication Password and Re-Enter Authentication Password fields. You cannot change the password without selecting the box.
	Truststore File	Specify the name and path of the keystore file containing the trusted certificates used when the remote server is configured to provide server authentication. For example: c:\security\truststore. Leave this field empty when server authentication is not used.
	Set Mutual Authentication Parameters	Specify Show to set mutual authentication information.
	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.

Section	Field	Description
	Customize HTTP Request Header Fields	Select Show to enable customized header fields or select Hide to disable the feature. Each of the following fields is conditional, depending on if you select 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 Settings	Listening IP Address and Port	Specify the IP address of the server where the SOAP driver is installed and the port number that this driver listens on. If you imported a sample configuration file, this field contains the IP address and port that you specified in the wizard.
	(Conditional) Authentication ID	Specify the authentication ID of the remote server to validate incoming requests. If the remote server does not send an Authentication ID, leave this field empty. If you imported a sample configuration file, this field contains the IP address and port that you specified in the wizard.
	(Conditional) Authentication Password and Re-Enter Authentication Password	Specify the authentication password of the remote server to validate incoming requests if you entered an Authentication ID above. Otherwise, leave these fields empty.
	Remove Existing Password	Click the box to remove the existing password, then specify the new password in the Authentication Password and Re-Enter Authentication Password fields. You cannot change the password without selecting the box.
	KMO Name	Specify the KMO name to be used in eDirectory™. When the server is configured to accept HTTPS connections, this name becomes the KMO name in eDirectory. The KMO name is the name before the “-” (dash) in the RDN. Leave this field empty when a keystore file (see Keystore File below) is used or when HTTPS connections are not used.

Section	Field	Description
	Keystore File	Specify the keystore name and path to the keystore file. This file is used when the server is configured to accept HTTPS connections, Leave this field empty when a KMO name is used (see KMO name above) or when HTTPS connections are not used.
	Keystore Password	Specify the keystore file password used with the keystore file specified above when this server is configured to accept HTTPS connections. Leave this field empty when a KMO name is used or when HTTPS connections are not used.
	Server Key Alias	Specify a server key alias when this server is configured to accept HTTPS connections. Leave this field empty when a KMO name is used or when HTTPS connections are not used.
	Server Key Password	When this server is configured to accept HTTPS connections, this is the key alias password (not the keystore password). Leave this field empty when a KMO name is used (see above) or when HTTPS connections are not used.
	Heartbeat Interval in Seconds	Specify the heartbeat interval in seconds. Leave this field empty to turn off the heartbeat. For more information about the heartbeat, see the information in the *Identity Manager Administration Guide* (http://www.novell.com/documentation/dirxml20/admin/data/bnotjfg.html).

9 Click Apply, then OK.

Configuring Subscriber Settings

1 In iManager, click DirXML > Overview.

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

3 In the DirXML Driver Overview page, click the driver's icon again, then scroll to Subscriber Settings.

4 Specify the Subscriber settings as described in Step 8 on page 19.

5 Click Apply, then click OK.

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

If the remote Web service you are accessing allows HTTPS connections, you can configure the subscriber to take advantage of this capability. You need a trust store containing a certificate issued by the certificate authority that signed the server's certificate. See "Configuring the Publisher to Receive HTTPS Connections" on page 24 for an example.

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

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

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

   For example:

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

2. Configure the subscriber to use the trust store you created in Step 1 on page 23.

   2a. In iManager, click DirXML > Overview.

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

   2c. In the DirXML Driver Overview page, click the driver's icon again, then scroll to Subscriber Settings.

   2d. In the Truststore File setting, specify the path to the trust store you created in Step 1 on page 23.

3. Click Apply, then click OK.

Configuring the Subscriber to Use a Proxy

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

1. In iManager, click DirXML > Overview.
2. Locate the driver set containing the SOAP driver, then click the driver's icon to display the DirXML Driver Overview page.
3. In the DirXML Driver Overview page, click the driver's icon again, then scroll to Subscriber Settings.
4. In the Proxy Host and Port setting, specify the host and port of the proxy using the following format:

   host:port

5. Click Apply, then click OK.

Configuring Publisher Settings

1. In iManager, click DirXML > Overview.
2. Locate the driver set containing the SOAP driver, then click the driver's icon to display the DirXML Driver Overview page.
3. In the DirXML Driver Overview page, click the driver's icon again, then scroll to Publisher Settings.
4. Enter the Subscriber settings as described in Step 8 on page 19.
5. Click Apply, then click OK.

Configuring the Publisher to Receive HTTPS Connections

1. Create a server certificate in iManager.
   - **1a** Click Novell Certificate Server > Create Server Certificate.
   - **1b** Select the server object where the SOAP driver is installed and specify a certificate nickname.
   - **1c** Select the Standard creation method.
   - **1d** Click Next, then click Finish.
2. Export a self-signed certificate from the certificate authority in eDirectory.
   - **2a** Click eDirectory Administration > Modify Object.
   - **2b** Select your tree's certificate authority object, then click OK.

     It is usually found in the Security container and is named something like *TREENAME CA.Security*.
   - **2c** Click Certificate > Self Signed Certificate.
   - **2d** Click Export.
   - **2e** When asked if you want to export the private key with the certificate, click No.
   - **2f** Based on the client to be accessing the Web service, select either binary DER or Base64 format for the certificate.

     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, then save it to a known location on your computer.
3. Import the self-signed certificate into the client's trust store.

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

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

     For more information on keytool, see Keytool - Key and Certificate Management Tool (http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/keytool.html).
   - **3b** Type the following command at a command prompt:

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

For example:

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

4 Configure the Publisher to use the server certificate you created in Step 1 on page 24.

4a In iManager, click DirXML > Overview.

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

4c In the DirXML 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 on page 24.

5 Click Apply, then click OK.

Creating XSLT Style Sheets

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

For detailed information on writing style sheets to handle other document types, refer to the sample style sheets that come with this driver. [insert link to style sheet basics in IM guide]

Using <operation-data> with Style Sheets

The driver shim applies special handling to subscriber commands based on the <operation-data> element. Operation data can be used to match commands with the responses they generate, which is useful for creating associations, or it can be used to override default Subscriber channel connection attributes.

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

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

To see how operation data is used to match commands with their responses, see the input and output transformation style sheets provided with the sample configurations for this driver.

To use operation data to override the default subscriber connection parameters:

1 Edit the subscriber settings section of the driver configuration.

2 Using the XML edit feature of iManager, find each subscriber setting that ends with a dash and the number 1, such as subURL-1, duplicate it, and increment the number.

For example, **subURL-2**.

3 Edit the values of the new settings to be the values you want to use for the second connection.

You can configure any number of connections this way as long as the numbers you use are incremental without gaps.

4 Add an attribute to the <operation-data> element called **connection** and give it the value of the connection number you want to use.

For example:

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

5 Using the Driver

After you complete the driver installation and import a sample configuration file, you must complete the following tasks:

- “Starting the Driver” on page 27
- “Migrating and Resynchronizing Data” on page 27
- “Activating the Driver” on page 28

Starting the Driver

1. In iManager, click DirXML > DirXML Overview.
2. Browse to and select the driver set where the driver exists.
3. In the SOAP driver icon, click inside the circle in the upper right corner of the icon to display the drop-down list.
4. Click Start Driver.

To further configure startup options, see “Configuring Driver Settings” on page 19.

Migrating and Resynchronizing Data

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

- **Migrate Data from eDirectory:** Allows you to select containers or objects you want to migrate from the Identity Vault to an application. When you migrate an object, the Identity Manager engine applies all of the Matching, Placement, and Create policies, as well as the Subscriber filter, to the object.
- **Migrate Data into eDirectory:** Assumes that the remote application (usually a Web Service) can be queried for entries that match the criteria in the publisher filter. However, because of the general nature of the SOAP driver the method for querying the Web Service (if there is one) is not known to the driver shim. Therefore, this feature does not usually work with the SOAP driver.
- **Synchronize:** The Identity Manager engine looks in the Subscriber class filter and processes all objects for those classes. Associated objects are merged. Unassociated objects are processed as Add events.

To use one of the options explained above:

1. In iManager, click DirXML > DirXML Overview.
2. Locate the driver set containing the SOAP driver, then click the driver icon.
3. Click the appropriate migration button.

Activating the Driver

You must activate the driver within 90 days of installation or the driver stops working.

For activation information, refer to "Activating Novell Identity Manager Products" (http://www.novell.com/documentation/dirxml20/index.html?page=/documentation/dirxml20/admin/data/afbx4oc.html) in the *Novell Nsure Identity Manager 2 Administration Guide* (http://www.novell.com/documentation/dirxml20/index.html?page=/documentation/dirxml20/admin/data/front.html#bktitle).

6 Troubleshooting the Driver

This section contains the following information on error messages:

- "Driver Shim Errors" on page 29
- "Java Customization Errors" on page 31

Driver Shim Errors

The following table identifies errors that might occur in the core driver shim. Error messages that contain a numerical code can have various messages depending on the application or Web service.

Error Message	Level	Description
307 Temporary Redirect	Retry	The Subscriber channel attempted to send data to the application or Web service but received a 307 Temporary Redirect response. The Subscriber waits for a period of time (usually 30 seconds) and tries again.
408 Request Timeout	Retry	The Subscriber channel attempted to send data to the application or Web service but received a 408 Request Timeout response. The Subscriber waits for a period of time (usually 30 seconds) and tries again.
503 Service Unavailable	Retry	The Subscriber channel attempted to send data to the application or Web service but received a 503 Service Unavailable response. The Subscriber waits for a period of time (usually 30 seconds) and tries again.
504 Gateway Timeout	Retry	The Subscriber channel attempted to send data to the application or Web service but received a 504 Gateway Timeout response. The Subscriber waits for a period of time (usually 30 seconds) and tries again.

Error Message	Level	Description
Various numeric error codes not listed above.	Error	HTTP servers, such as those the Subscriber channel might be communicating with, return numeric values and a short descriptive message to indicate the status of the request. Numbers in the range of 200-299 indicate success, so an error message isn't generated. Numbers listed above (307, 408, 503, and 504) indicate temporary conditions, so the request is retried. Other numerical error codes result in an error message containing that code and the message provided by the HTTP server. In most cases, the driver continues to run, and the command that caused the error isn't retried.
Problem communicating with HTTP server. Make sure server is running and accepting requests.	Retry	The Subscriber channel received an IOException while communicating or attempting to communicate with the HTTP server. You might receive this error because the server is either not running, is overloaded, cannot be accessed because of firewall or other restrictions, or the URL provided in the subscriber configuration is not correct. The command that caused this error is retried later.
The HTTP/SOAP driver doesn't return any application schema by default. If there is application specific schema you want the shim to report, you can write your own Java class that implements the SchemaReporter interface and then configure the driver to load your class as a Java extension. See the driver documentation for more details.	Warning	The DirXML engine called the DriverShim.getSchema() method of the driver, and the driver has not been extended with a SchemaReporter customization. The driver continues to run.
Subscriber.execute() was called but the Subscriber was not configured correctly. The command was ignored. You should either configure the Subscriber or clear the Subscriber's filter so it doesn't receive commands.	Warning	The Subscriber channel of the driver isn't initialized properly. The most likely cause is an improperly formatted driver configuration. The driver continues to run but displays this message each time an event is received by the Subscriber channel.
pubHostPort must be in the form host:port	Fatal	An error occurred with the Publisher channel configuration. Review the Publisher channel parameters to verify that both a valid host and a valid port number are provided.
MalformedURLException	Fatal	The URL supplied in the Subscriber channel parameters isn't in a valid URL format.
Multiple Exceptions	Fatal	This message appears in the trace when the HTTP listener fails to properly initialize. This can occur for a variety of reasons. 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.

Error Message	Level	Description
HTTPS Hostname Wrong: Should Be . . .	Retry	This message appears when an SSL handshake fails on the Subscriber channel. It indicates that the subject presented with the server certificate doesn't match the IP address or hostname given in the HTTPS URL. Use a DNS hostname rather than an IP address in the URL.

Java Customization Errors

The following table identifies errors that might occur in the customized Java extensions.

Message	Level	Description
SchemaReporter init problem: *extension-specific message*	Fatal	The SchemaReporter Java customization had a problem initializing. The driver shuts down.
Extension (custom code) init problem: *extension specific message*	Fatal	One of the following Java extensions failed to initialize: ◆ SubscriberTransport ◆ PublisherTransport ◆ DocumentModifiers ◆ ByteArrayModifiers The driver shuts down.
Various other errors	Varies	The interfaces provided for Java extensions return error messages on the trace screen and sometimes to the Identity Manager engine. Sometimes it is difficult to distinguish errors of this type from other errors that originate in the core driver shim. If you get errors that are not listed in this table and you are using Java extensions, check with whomever provided you with the extensions for a list of error codes for that particular extension.

A Using Java Extensions

The functionality of the Nsure™ Identity Manager Driver for SOAP can be extended by using Java*.Using an API defined by Java interfaces, you can create your own custom Java classes that have access to the data passing through the Subscriber channell 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" on page 33
- "Creating and Configuring Java Extensions" on page 34

Overview

If the application you are using with the Identity Manager Driver for SOAP uses non-XML data, you can create Java extensions to convert the non-XML data to XML data. Or, you may want to change various protocols, including XML and HTTP. For example, the defalult 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.

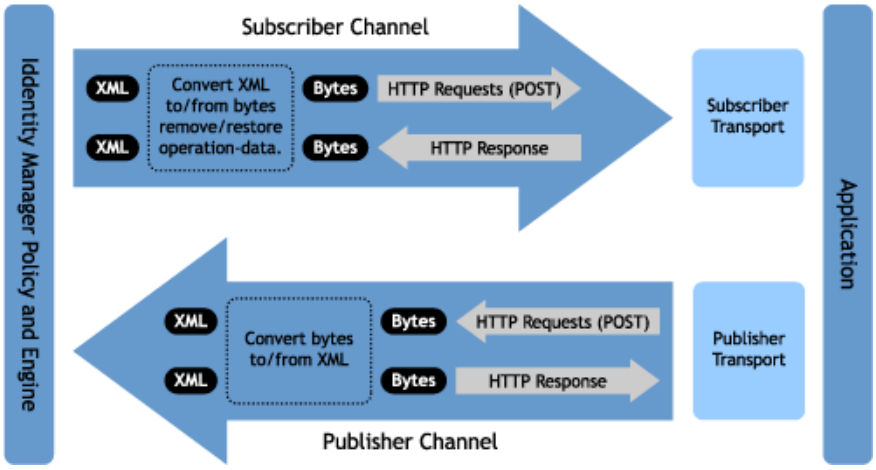
As illustrated in the following diagram, there are eleven points where functionality can be extended:

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

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

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

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



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 if desired the commands and events passing through the driver shim. 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 Novell's Developer Downloads Web site (http://developer.novell.com/ndk/downloadaz.htm) as a guide, write the Java code for your class. In the A-Z listing, search for SOAP Driver. You should name your class using any Java package and class name that is convenient to your environment and your organization.

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

1. Prepare your environment.

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

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

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

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

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

4. Put a copy of the required jar files in a convenient location like the root of your compile directory just outside the com directory, then access a system command prompt or shell prompt with that location as the current directory.
5. Compile your class by entering one of the following:
   - For Windows: **`javac -classpath SOAPUtil.jar;nxsl.jar com\novell\idm\*.java`**
   - For Linux or UNIX: **`javac -classpath SOAPUtil.jar:nxsl.jar com/novell/idm/*.java`**
6. Create a Java archive file containing your class by entering one of the following:
   - For Windows: **`jar cvf mydriverextensions.jar com\novell\idm\*.class`**
   - For Linux or UNIX: **`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 Extensions, 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 vlue for Init Parameter.

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

You can now use your custom class.