AUTHORIZED DOCUMENTATION

Driver for Delimited Text Implementation Guide

Novell®
Identity Manager

3.6

July 23, 2008

www.novell.com

Legal Notices

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

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

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

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

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

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

Novell Trademarks

For a list of Novell trademarks, see Trademarks (http://www.novell.com/company/legal/trademarks/tmlist.html).

Third-Party Materials

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

Contents

About This Guide

This guide explains how to install and configure the Identity Manager Driver for Delimited Text (Delimited Text driver).

Audience

This guide is for Novell® eDirectory™ and Identity Manager administrators who are using the Delimited Text driver.

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. 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.

Documentation Updates

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

Additional Documentation

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

Documentation Conventions

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

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

Overview

1

The Delimited Text driver lets you use delimited text files (.csv or .xml) to synchronize data between the Identity Vault and applications. Essentially, you can transfer data from the Identity Vault to any system that can consume delimited text files, or transfer data to the Identity Vault from any system that can generate delimited text files. The following sections provide information to help you understand the Delimited Text driver.

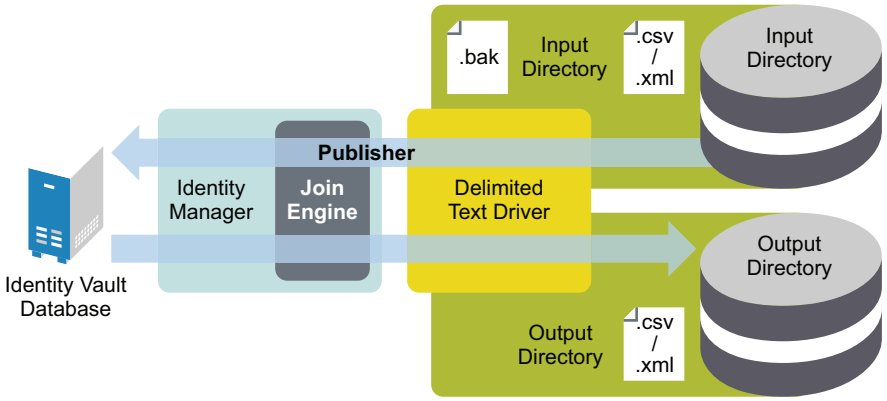
1.1 How the Delimited Text Driver Works

The Delimited Text driver uses the Publisher channel, the Subscriber channel, and policies to control data flow between the Identity Vault and the delimited text files, as explained in the following sections:

1.1.1 Publisher and Subscriber Channels

The Delimited Text driver provides data flow along the Publisher and Subscriber channels as shown in the following diagram.

Figure 1-1 *Data Flow*



The example configuration that ships with this driver includes both Subscriber and Publisher channels. However, in many configurations, only one-way data flow is required. In those configurations, only a Publisher or Subscriber channel is used. The other channel is disabled. For more information, see Chapter 5, "Setting Up One-Way Synchronization," on page 25.

Publisher Channel

The Publisher channel reads information from input text files on your local file system and submits that information to the Identity Vault via the Metadirectory engine.

By default, the Publisher channel does the following:

1. Checks the input directory every 10 seconds.
2. Processes any files that have a `.csv` extension.
3. Changes `.csv` extensions of processed files to `.bak`.
4. Cycles through this process until you stop the driver.

Subscriber Channel

The Subscriber channel watches for additions and modifications to Identity Vault objects and creates output files on your local file system that reflect those changes.

By default, the Subscriber channel keeps an output file open until either 200 transactions have been logged or 30 seconds have elapsed. When either of these thresholds is reached, the output file is saved with a *`number`*`.csv` filename and a new output file is opened.

1.1.2 Policies

Policies control data synchronization between the driver and the Identity Vault. The following table provides information on the set of preconfigured policies that come with the Delimited Text driver. For information about modifying policies, see *Policies in iManager for Identity Manager 3.6* or *Policies in Designer 3.0*.

Table 1-1 *Preconfigured Policies*

Policy	Description
Schema Map	Configured on the driver object. Maps Identity Vault User properties to application attributes as follows: Surname > LastName Given Name > FirstName Title > Title Internet EMail Address > Email Telephone Number > WorkPhone Facsimile Telephone Number > Fax mobile > WirelessPhone Description > Description The application attributes correspond to the sequence of values in the file or, if present, to the attributes associated with unnamed XDS <`field`> elements.

Policy	Description
Input Transform	Configured on the driver object. If the input document is an XML document, no transformations are made. If the document is a delimited text file, each record is transformed into an XDS Add element for User objects with attributes defined by the schema map. The user CN is formed by concatenating the values of first name and last name. Associations are defined by value the user's e-mail attribute.
Output Transform	Configured on the driver object. Specifies that a comma is used as the delimiter character for output files and that the file format is comma-separated value (CSV) format.
Create	Configured on the Publisher channel. Specifies that in order for a User to be created in an Identity Vault, the Given Name and Internet EMail Address attributes must be defined.
Matching	Configured on the Publisher channel. Specifies that a user in an Identity Vault is the same user specified in the input file when the value of Internet Email Address is the same in both places. In the case of a match, only changed attributes are updated in the Identity Vault.
Placement	Configured on the Publisher channel. Specifies that a new user is placed in the Users or Active container and named with the CN created by the Input Transform rule. You need to create a Users\Active container at the root of your tree before you start the driver.
Event Transform	Configured on the Subscriber channel. If an Identity Vault reports a Modify or Sync event, those events are changed to an instance element that can be used to create a complete output record.

1.1.3 Delimited Text File Types

The driver currently supports two types of files:

- “Comma-Separated Values Files” on page 11
- “XML Files in XDS Format” on page 12

Comma-Separated Values Files

Comma-separated value (CSV) files are text files that contain data divided into fields and records. Fields are delimited by commas, and records are delimited by a hard return.

If you need a comma or hard return within the value of a particular field, the entire field value should be enclosed in quotes.

Because the meaning of each field in a CSV file is derived from its position, each record in a CSV file should have the same number of fields. Field values can be left blank, but each record should have the same number of delimiter characters.

XML Files in XDS Format

The XDS format is the defined Novell® subset of possible XML formats. This is the initial format for data coming from an Identity Vault. By modifying default rules and changing the style sheets, the Delimited Text driver can be configured to work with any XML format.

For detailed information on the XDS format, refer to NDS DTD Commands and Events (http://developer.novell.com/ndk/doc/dirxml/index.html?dirxmlbk/data/a5323rs.html).

For information on configuring the driver to use XML files in the XDS format, see Chapter 6, "Configuring for XDS XML Files," on page 27.

1.2 Java Interfaces to the Driver

The Delimited Text driver includes four Java* interfaces that enable you to add extensions, which are optional. These enhancements to the driver require Java programming. For more information, see Chapter 8, "Using Java Interfaces to Customize File Processing," on page 31.

1.3 Local and Remote Platforms

The Delimited Text driver runs on the Metadirectory server or uses the Remote Loader to run on another server.

1.4 Entitlements

The Delimited Text driver does not implement entitlements.

1.5 Password Synchronization

The Delimited Text driver has policies to handle password synchronization. However, no automatic password synchronization exists for the Delimited Text driver. You must be aware of and decide on which attribute you want to hold the password, then synchronize that attribute. Any password synchronization for the driver is just an attribute synchronization.

Installing Driver Files

2

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

Unless you extend the driver's functionality along with the Java interfaces (see Chapter 8, "Using Java Interfaces to Customize File Processing," on page 31), the driver is capable of only reading input text files from the local file system of the server where the driver is running. This means that your input directory must be located on the same server as the driver. You have three options:

- If your input directory can be located on the Metadirectory server and the Delimited Text driver files are already installed on the server, skip this section and continue with Chapter 3, "Creating a New Driver," on page 15/
- If your input directory can be located on the Metadirectory server but the driver files are not already installed on the server, install the files by using the instructions in "Installing the Metadirectory Server" in the *Identity Manager 3.6 Installation Guide*.
- If your input directory cannot be located on the Metadirectory server, install the Remote Loader (required to run the driver on a non-Metadirectory server) and the driver files on the server where the input directory resides. See "Installing the Remote Loader" in the *Identity Manager 3.6 Installation Guide*.

Creating a New Driver

3

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

- Section 3.1, "Preparing Data Locations," on page 15
- Section 3.2, "Creating the Driver in Designer," on page 15
- Section 3.3, "Creating the Driver in iManager," on page 18
- Section 3.4, "Activating the Driver," on page 21

3.1 Preparing Data Locations

You need to make sure that the driver's input and output directories exist. The input directory is where the driver looks for delimited text files to be processed into the Identity Vault. The output directory is where the driver places delimited text files to be processed by the target application.

The input and output directories must be located on the same server as the driver. The directories can have any names supported by the server operating system.

3.2 Creating the Driver in Designer

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

- Section 3.2.1, "Importing the Driver Configuration File," on page 15
- Section 3.2.2, "Configuring the Driver Settings," on page 16
- Section 3.2.3, "Deploying the Driver," on page 17
- Section 3.2.4, "Starting the Driver," on page 17

3.2.1 Importing the Driver Configuration File

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

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

   Output File Path: Specify the platform-specific path to the local directory where the driver will create output files. This directory must already exist (see Section 3.1, "Preparing Data Locations," on page 15).

Input File Path: Specify the platform-specific path to the local directory where input files are placed. This directory must already exist (see Section 3.1, "Preparing Data Locations," on page 15).

User Container: Select the Identity Vault container where any new users created from delimited text file information will be placed. This value becomes the default for all drivers in the driver set. If you don't want to change this value for all drivers, leave this field alone and change the value on the driver's Global Configuration Values page after you've finished importing the driver.

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

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

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

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

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

6 Click *Next* to import the driver configuration.

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

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

or

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

3.2.2 Configuring the Driver Settings

After importing the driver configuration file, the Delimited Text driver will run. However, the basic configuration might not meet the requirements for your environment. For example, you might need to change the following settings:

- How often the driver polls the input directory for new files.
- The file extension used to designate an input file or output file.
- The fields included in input and output files.
- The character used as the text delimiter in input and output files.

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. These settings let you control the format and content of the input and output files.

The driver configuration settings are explained in Appendix A, “Driver Properties,” on page 39.

If you do not have the Driver Properties page displayed in Designer:

1. Open your project.
2. In the Modeler, right-click the driver icon  or the driver line, then select *Properties*.

3.2.3 Deploying the Driver

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

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

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

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

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

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

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

   8b Click *OK*.

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

   8d Click *OK*.
9. Click *OK*.

3.2.4 Starting the Driver

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

To start the driver:

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

For information about management tasks with the driver, see Chapter 9, "Managing the Driver," on page 35.

3.3 Creating the Driver in iManager

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

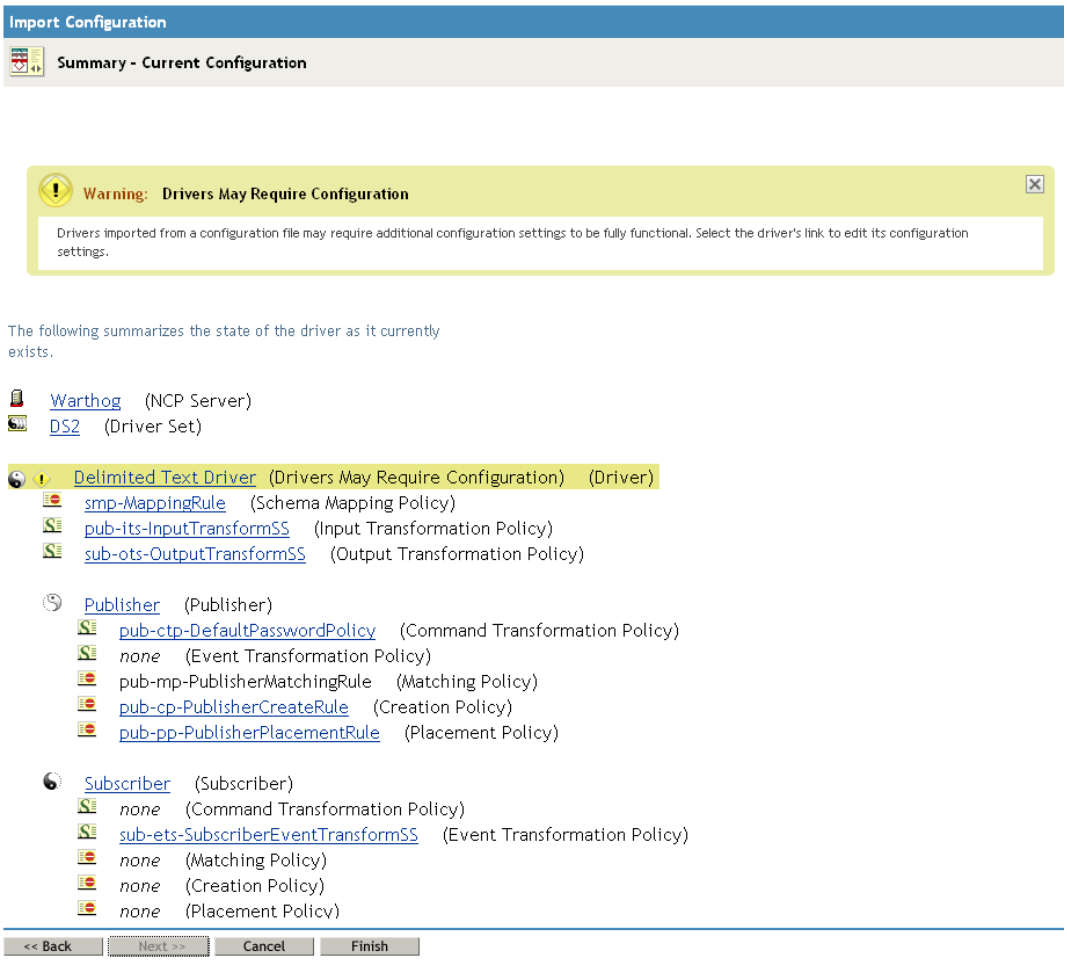
3.3.1 Importing the Driver Configuration File

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

Prompt	Description
Where do you want to place the new driver?	You can add the driver to an existing driver set, or you can create a new driver set and add the driver to the new set. If you choose to create a new driver set, you are prompted to specify the name, context, and server for the driver set.
Import a configuration into this driver set	Use the default option, *Import a configuration from the server (.XML file)*. In the *Show* field, select *Identity Manager 3.6 configurations*. In the *Configurations* field, select the Delimited Text CSV sample file.
Driver name	Type a name for the driver. The name must be unique within the driver set.
Output File Path	Specify the platform-specific path to the local directory where the driver will create output files. This directory must already exist (see Section 3.1, "Preparing Data Locations," on page 15).
Input File Path	Specify the platform-specific path to the local directory where input files are placed. This directory must already exist (see Section 3.1, "Preparing Data Locations," on page 15).

Prompt	Description
User Container	Select the Identity Vault container where any new users created from delimited text file information will be placed. This value becomes the default for all drivers in the driver set. If you don't want to change this value for all drivers, leave this field alone and change the value on the driver's Global Configuration Values page after you've finished importing the driver.
Driver is Local/Remote	Select *Local* if this driver will run on the Metadirectory server without using the Remote Loader service. Select *Remote* if you want the driver to use the Remote Loader service, either locally on the Metadirectory server or remotely on another server.
Remote Host Name and Port	This applies only if the driver is running remotely. Specify the host name or IP address of the server where the driver's Remote Loader service is running.
Driver Password	This applies only if the driver is running remotely. Specify the driver object password that is defined in the Remote Loader service. The Remote Loader requires this password to authenticate to the Metadirectory server.
Remote Password	This applies only if the driver is running remotely. Specify the Remote Loader's password (as defined on the Remote Loader service). The Metadirectory engine (or Remote Loader shim) requires this password to authenticate to the Remote Loader
Define Security Equivalences	The driver requires rights to objects within the Identity Vault and to the input and output directories on the server. The Admin user object is most often used to supply these rights. However, you might want to create a DriversUser (for example) and assign security equivalence to that user. Whatever rights that the driver needs to have on the server, the DriversUser object must have the same security rights.
Exclude Administrative Roles	You should exclude any administrative User objects (for example, Admin and DriversUser) from synchronization.

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



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

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

or

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

3.3.2 Configuring the Driver Settings

After importing the driver configuration file, the Delimited Text File driver will run. However, the basic configuration might not meet the requirements for your environment. For example, you might need to change the following settings:

- How often the driver polls the input directory for new files.
- The file extension used to designate an input file or output file.
- The fields included in input and output files.
- The character used as the text delimiter in input and output files.

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).

To configure the settings:

1. Make sure the Modify Object page for the Delimited Text driver is displayed in iManager. If it is not:
   - **1a** In iManager, click  to display the Identity Manager Administration page.
   - **1b** Click *Identity Manager Overview*.
   - **1c** Browse to and select the driver set object that contains the new driver.
   - **1d** Click the driver set name to access the Driver Set Overview page.
   - **1e** Click the upper right corner of the driver, then click *Edit properties*.
2. Review the settings on the various pages and modify them as needed for your environment. The configuration settings are explained in Appendix A, "Driver Properties," on page 39.

   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. These settings let you control the format and content of the input and output files.
3. After modifying the settings, click *OK* to save the settings and close the Modify Object page.
4. (Conditional) If the Delimited Text driver's Summary page for the Import Configuration wizard is still displayed, click *Finish*.

   WARNING: Do not click *Cancel* on the Summary page. This removes the driver from the Identity Vault and results in the loss of your work.

3.3.3 Starting the Driver

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

To start the driver:

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

For information about management tasks with the driver, see Chapter 9, "Managing the Driver," on page 35.

3.4 Activating the Driver

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

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

Upgrading an Existing Driver

4

If you are running the driver on the Metadirectory server, the driver shim files are updated when you update the server unless they were not selected during a custom installation. If you are running the driver on another server, the driver shim files are updated when you update the Remote Loader on the server.

The 3.6 version of the driver shim supports drivers created by using any 3.*x* version of the driver configuration file. You can continue to use these driver configurations until you want to upgrade them.

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

4.1 Supported Upgrade Paths

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

4.2 What's New in Version 3.6

Version 3.6 of the driver does not include any new features that are not already in the 3.5.1 version. However, the process for creating a new driver has changed. For detailed information, see Chapter 3, "Creating a New Driver," on page 15.

4.3 Upgrade Procedure

The process for upgrading the Delimited Text driver is the same as for other Identity Manager drivers. For detailed instructions, see "Upgrading" in the *Identity Manager 3.6 Installation Guide*.

Setting Up One-Way Synchronization

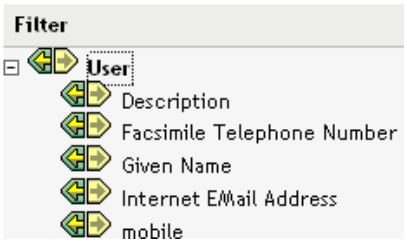
5

If your data synchronization goes only one way, disable the channel that you won't use. To disable one of the channels, clear the filters on the channel you don't need and remove the path for the input or output directory, depending on the channel.

For example, if you only need a Publisher channel:

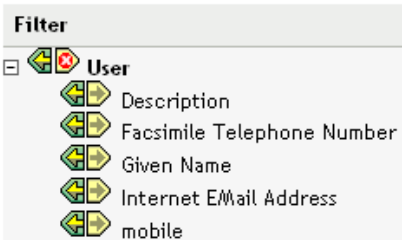
1 On the Filter editor in iManager, clear the filters on the Subscriber channel.

1a For example, select the *User* class.



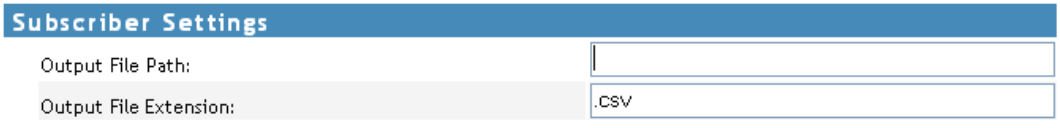
1b Select *Ignore* in the Subscribe section.

The Subscriber channel icon for the User class changes to show that the class is no longer being synchronized.



2 Click *OK* to save the changes.

3 Edit the driver properties to remove the path specified for the *Output File Path*. This setting is located under the *Subscriber Settings* for the *Driver Parameters* on the *Driver Configuration* page. For details, see Section A.1.5, "Driver Parameters," on page 42.



If you only need a Subscriber channel, clear the filters on the Publisher object and remove the path specified for the *Input File Path* in the *Driver Parameters* section.

Configuring for XDS XML Files

6

You can use XML files in XDS format instead of comma-separated value (CSV) files with the driver.

Because you generally use this driver only with a Publisher or Subscriber channel, perform only the steps from the section that you need.

6.1 Using the Publisher Channel

To have the driver accept input in XML format, change the input file extension to `.xml`.

6.2 Using the Subscriber Channel

To have the driver send output in XDS format, remove the Event Transform and Output Transform style sheets from the Subscriber channel.

1. In iManager, select *eDirectory Administration* > *Delete Object*.
2. Browse to the driver’s Subscriber object, then select the SubscriberEventTransformSS object.
3. Click *OK*.
4. Click *Repeat Task*.
5. Browse to and select the driver’s OutputTransformSS object.
6. Click *OK* twice.

Using Style Sheets to Configure Data Synchronization

7

The real power of Identity Manager is in managing the shared data itself. This section covers some common customizations for the Delimited Text driver.

The example configuration available with the driver uses comma-separated value files. However, you can use the driver in many ways. It is designed to be as flexible as possible. The driver passes the text-based files largely unchanged to the style sheets. The style sheets do most of the work. You can write new style sheets to allow the driver to work with almost any text-based file that contains predictably repeatable patterns.

The basis for this exchange is the <delimited-text> XML element. For example, to design a Publisher channel that reads information from a text file, create an Input Transform style sheet that receives the contents of the file and converts it into a <delimited-text> element.

The following is an example of a <delimited-text> element:

```
<delimited-text>
     <record>
          <field>John</field>
          <field>Maxfield</field>
          <field>555-1212</field>
     </record>
     <record>
          <field>Sarah</field>
          <field>Lopez</field>
          <field>555-3434</field>
     </record>
</delimited-text>
```

When field elements appear like this without an identifying name attribute, the driver uses the field position and matches it with the position of the Field Name driver parameter.

You can provide the field name within the XML:

```
<delimited-text>
     <record>
          <field name="FirstName">John</field>
          <field name="LastName">Maxfield</field>
          <field name="Phone">555-1212</field>
     </record>
     <record>
          <field name="FirstName">Sarah</field>
          <field name="LastName">Lopez</field>
          <field name="Phone">555-3434</field>
     </record>
</delimited-text>
```

For detailed information on writing style sheets to handle other document types, refer to the sample style sheets that come with this driver. If you create the driver by using the example configuration, you can use Input Transform, Output Transform, and Event Transformation style sheets as a starting point.

Using Java Interfaces to Customize File Processing

8

Java interfaces enable you to customize file processing by using Java classes that you write:

- InputSorter
- InputSource
- PreProcessor
- PostProcessor

These enhancements to the driver require Java programming. To implement this functionality, complete the following processes:

- Create a Java class that implements one of the new interfaces
- Create a Java `.jar` file that contains your new class
- Configure the driver to use the new class

These interfaces enable you to add extensions, which are optional. The driver continues to function as before without extensions. However, if you want to directly modify the behavior of the driver, but have been unable to make these modifications from a style sheet or DirXML® Script, extending the Delimited Text driver can be useful.

By using Java classes that you write, you can use the interfaces to customize the publish and subscribe processes in the following ways:

Table 8-1 *Customizing the Publish and Subscribe Processes*

Process	Interface	Description
Publish	InputSorter	Defines the processing order of multiple input files. The system where your driver is installed determines the default processing order. For example, files on an NT system are processed in alphabetical order. You can use the InputSorter to impose the processing order that you require.
Publish	InputSource	Provides data other than the files in the default location for the driver to process. For example, you could check an FTP server for input files and then transfer the files to the local file system for processing.

Process	Interface	Description
Publish	PreProcessor	Ties data manipulation required to prepare input files for driver processing directly to the driver. Before this interface was available, preprocessing was independent of the driver. You could create a separate application that monitored another directory for input files, modify the files in some way, and then copy the files to the input directory of the driver. By creating a class that implements the PreProcessor, you can do this type of preprocessing more directly.
Subscribe	PostProcessor	Ties data manipulation required by the application consuming Identity Vault output directly to the driver.

8.1 Creating a Java Class

JavaDoc and an example class are included with the driver to help you implement this functionality. Find the JavaDoc at `nt\dirxml\drivers\delimitedtext\javadocs` and the sample code at `nt\dirxml\drivers\delimitedtext\extensions\sample code`.

8.2 Creating a Java .jar File

After you have implemented your class file, create a Java `.jar` file (Java archive) using the jar tool. The `.jar` file must contain the class that you have created. Put the `.jar` file into the `novell/nds/lib` directory. The path might differ, depending on the platform you're on, but it should be the same location as `DelimitedTextShim.jar` and `DelimitedTextUtil.jar`.

8.3 Configuring the Driver to Use the New Class

After you have placed the new `.jar` file in the correct location, configure the driver to use your new class by modifying the driver's properties.

1. In iManager, open the driver's property page. To do so:
   - **1a** In iManager, click  to display the Identity Manager Administration page.
   - **1b** In the *Administration* list, click *Identity Manager Overview*.
   - **1c** Open the driver set that contains the driver.
   - **1d** Click the driver icon to open the *Identity Manager Driver Overview* page.
   - **1e** Click the upper-right corner of the driver icon to open the Actions menu, then click *Edit properties*.
2. Select *Driver Configuration*.
3. Scroll to *Driver Parameters*, then click *Edit XML*.
4. Locate the `<publisher-options>` section of the file.

   This file defines which parameters and values appear in the Driver Parameters section of the D*river Configuration* page.

   For each class you created that works on the Publisher channel, you enter an additional option in the `<publisher-options>` section. After you've updated this file, you'll see your new options in the interface.

5 For each new class you created on the Publisher channel, add an entry corresponding to the interface type. Use the following table as a guide:

Interface	New Entry

InputSorter	`<input-sorter display-name="InputSorter Class">com.acme.MyNewClass</input-sorter>`
	
	`<input-sorter-params display-name="InputSorter init string">MY CONFIG PARAMS</input-sorter-params>`

InputSource	`<input-source display-name="InputSource Class">com.acme.MyNewClass</input-source>`
	
	`<input-source-params display-name="InputSource init string">MY CONFIG PARAMS</input-source-params>`

PreProcessor	`<pre-processor display-name="PreProcessor Class">com.acme.MyNewClass</pre-processor>`
	
	`<pre-processor-params display-name="PreProcessor init string">MY CONFIG PARAMS</pre-processor-params>`

5a Replace `com.acme.MyNewClass` with the name of the class that you have defined along with a full package identifier.

5b Replace `MY CONFIG PARAMS` with any information that you want to pass to the init method of your class.

The init method of your class is then responsible for parsing the information contained in this string. If your class doesn't require a configuration string to be passed to the init method, you can leave off the whole element, in which case null would be passed to the init method.

6 If you created a PostProcessor rule, locate the `<subscriber-options>` section of the file and add the following lines:

```
<post-processor display-name="PostProcessor Class">com.acme.MyNewClass</post-processor> <post-processor-params display-name="PostProcessor init string">MY CONFIG PARAMS</post-processor-params>
```

6a Replace `com.acme.MyNewClass` with the name of the class that you have defined along with full package information.

6b Replace `MY CONFIG PARAMS` with any information that you want to pass to the init method of your class.

The init method of your class is then responsible for parsing the information contained in this string. If your class doesn't require a configuration string to be passed to the init method, you can leave off the entire element, in which case null would be passed to the init method.

7 Click *OK*.

Managing the Driver

9

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

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

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

Troubleshooting

10

Viewing driver processes is necessary to analyze unexpected behavior. To view the driver processing events, use DSTrace. You should only use it during testing and troubleshooting the driver. Running DSTrace while the drivers are in production increases the utilization on the Identity Manager server and can cause events to process very slowly. For more information, see "Viewing Identity Manager Processes" in the *Identity Manager 3.6 Common Driver Administration Guide*.

Driver Properties

A

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

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

A.1 Driver Configuration

In iManager:

1. In iManager, click  to display the Identity Manager Administration page.
2. Open the driver set that contains the driver whose properties you want to edit. To do so:
   - **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 Delimited Text driver icon, then click the upper right corner of the driver icon to display the *Actions* menu.
4. Click *Edit Properties* to display the driver's properties page.

In Designer:

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

The Driver Configuration options are divided into the following sections:

A.1.1 Driver Module

The Driver Module section lets you change the driver from running locally to running remotely or the reverse.

Option	Description
Java	Used to specify the name of the Java class that is instantiated for the shim component of the driver. This class can be located in the `classes` directory as a class file, or in the `lib` directory as a `.jar` file. If this option is selected, the driver is running locally. The name of the Java class is: `com.novell.nds.dirxml.driver.delimitedtext.DelimitedTextDriver`
Native	Used to specify the name of the `.dll` file that is instantiated for the application shim component of the driver. If this option is selected, the driver is running locally.
Connect to Remote Loader	Used when the driver is connecting remotely to the connected system. Designer includes two suboptions: • *Driver Object Password*: Specifies a password for the Driver object. If you are using the Remote Loader, you must enter a password on this page. Otherwise, the remote driver does not run. The Remote Loader uses this password to authenticate itself to the remote driver shim. • *Remote Loader Client Configuration for Documentation*: Includes information on the Remote Loader client configuration when Designer generates documentation for the Delimited Text driver.

A.1.2 Driver Object Password (iManager Only)

Option	Description
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.

A.1.3 Authentication

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

Option	Description
Authentication information for server	Displays or specifies the server that the driver is associated with.
Authentication ID	Specify a user application ID. This ID is used to pass Identity Vault subscription information to the application. Example: `Administrator`

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

A.1.4 Startup Option

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

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

A.1.5 Driver Parameters

The Driver Parameters section lets you configure the driver-specific parameters. When you change driver parameters, you tune driver behavior to align with your network environment. For example, you might find the default Publisher polling interval to be shorter than your synchronization requires. Making the interval longer could improve network performance while still maintaining appropriate synchronization.

Parameter	Description
Driver parameters for server	Displays or specifies the server name or IP address of the server whose driver parameters you want to modify.
Edit XML	Opens an editor so that you can edit the driver's configuration file.
Driver Options	
Field Delimiter	Specifies the character that is used to delimit field values in the input files. It must be one character. The default is a comma. If the values of any of the input fields contain this character, enclose the entire value in quotes to prevent it from being seen as a delimiter. Changing this delimiter parameter to something other than a comma does not automatically change the delimiter character used in the output files when a Subscriber is used. To change the delimiter character in the output files, edit the Output Transform style sheet. The delimiter character is assigned to a variable near the top of that style sheet.
Field Names	Specifies a comma-separated list of attribute names that can be referred to in the Schema Mapping rule. In the input files, the fields of the records must correspond to the order and positioning of the names in this list. For example, if you list eight field names in this parameter, each record of the input files should have eight fields separated by the field delimiter character. On Windows*, see `sample.csv` in the `delimitedtext/samples` directory for an example. On Solaris* and Linux*, `sample.csv` is located in the `/usr/lib/dirxml/rules/delim` directory. The default values are LastName, FirstName, Title, Email, WorkPhone, Fax, WirelessPhone, and Description.
Object Class Name	Specifies the Novell® eDirectory™ class name that should be used when creating new objects to correspond to input files.

Parameter	Description
Allow Driver to Consume Its Own Output?	Prevents you from inadvertently creating a situation in which the driver writes output files that are immediately read in again as input of the same driver. The default is *No*. By default, the driver won't load if all the following conditions occur: ◆ You have both a Subscriber channel and a Publisher channel. ◆ The input and output directories are the same. ◆ The input and output file extensions are the same. If you want to feed the output of the Subscriber channel into the input of the Subscriber channel as a way to detect Identity Vault events to trigger other changes in the Identity Vault, set this parameter to *Yes*. For example, to update the Full Name attribute when the Given Name, Surname, or Initials attributes are updated, set this parameter to *Yes*.
Subscriber Options	
Output File Path	Specifies the directory on the local file system where output files will be created. An error occurs if this directory doesn't exist. The default values are: Windows: `c:\csvsample\output` Solaris or Linux: `/csvsample/output`
Output File Extension	Output files have a unique name that ends with the characters in the *Output File Extension* parameter. If the output files from a Subscriber channel are used as input files for the Publisher channel of another Delimited Text driver, the destination file extension must match the source file extension parameter of the second driver.
Destination File Character Encoding (leave blank for default)	When this parameter contains no value, the default Java character encoding for your locale is used. To use an encoding other than the default for your locale, enter one of the canonical names from the Supported Encodings table (http://java.sun.com/j2se/1.4.2/docs/guide/intl/encoding.doc.html). The Publisher and Subscriber channels can use different character encodings.
Maximum Number of Transactions per Output File	Specifies the maximum number of transactions that are written to a single output file. When the file transaction limit is reached, the file closes, and a new file is created for subsequent transactions. To limit the number of transactions that can be written to a single file, leave this parameter blank or set it to zero. For more information, see the next item, *Maximum Time in Seconds Before Flushing All Transactions*.
Maximum Time in Seconds before Flushing All Transactions	If no new transactions have been written to the output file in the amount of time specified in this parameter, the file is closed. When new transactions need to be written, a new output file is created. If you don't want to limit the time that can pass before the output file is closed, leave this parameter blank or set it to zero.

Parameter	Description
Time of Day (Local Time) to Flush All Transactions	If a value is supplied for this parameter, the current output file is closed at the specified time each day. Subsequent transactions are written to a new file. This parameter does not prevent the *Maximum Number of Transactions per Output File* or the *Maximum Time in Seconds before Flushing All Transactions* parameters from also acting as output file thresholds. If you use this parameter and only want one file per day, set the other two parameters to zero. The format of this parameter can be HH:MM:SS (using the 24-hour clock) or H:MM:SS AM/PM. An hour is required, but the minutes and seconds are optional. Because the parameter assumes local time, any time zone information included in the value is ignored. The previous three parameters (*Maximum Number of Transactions per Output File*, *Maximum Time in Seconds before Flushing All Transactions*, and *Time of Day to Flush All Transactions*) are all capable of acting as a threshold for the transaction size a file is able to grow to, or for the time that it remains open to accept new transactions. As long as an output file is still open for writing by the Delimited Text driver, it shouldn't be considered as finalized. Avoid opening the file in any other process until the driver closes the file. For this reason, one of the three previous parameters must be set to assure that output files don't remain open indefinitely. To avoid this condition, if the driver detects that all three parameters are blank (or zero), it automatically sets the Maximum Number of Transactions per Output File to the value of 1.
Publisher Options	
Input File Path	The Publisher channel looks for new input files in the Input File Path, which is a directory on the local file system. Example paths: ◆ On Windows: `c:\csvsample\input` ◆ On Solaris and Linux: `/usr/lib/dirxml/rules/delim`
Input File Extension	The extension used to designate input files (for example, `csv`).

Parameter	Description
Source File Character Encoding (leave blank for default)	When this parameter contains no value, the default Java character encoding for your locale is used. To use an encoding other than the default for your locale, enter one of the canonical names from the Supported Encodings table (http://java.sun.com/j2se/1.4.2/docs/guide/intl/encoding.doc.html). If the Input File Extension parameter is `.xml`, the Source File Character Encoding can be indicated in one of two ways. ◆ If a value is indicated in the *Source File Character Encoding* parameter, it is used. ◆ If the parameter is blank, and if the XML document specifies an Encoding Declaration as described in the W3C XML Recommendation (http://www.w3.org/TR/REC-xml#charencoding) in paragraph 4.3.3, the Encoding Declaration is handled by the XML parser in the Metadirectory engine. The Identity Manager XML parser handles the following character encodings: ◆ UTF-8 ◆ UTF-16 ◆ ISO-8859-1 ◆ US-ASCII
Rename File Extension	The Publisher channel uses only files that have the extension specified in the parameter. After the files have been processed, the value of the *Rename File Extension* parameter is appended to the filename, so the Publisher channel won't try to process the same file again. If the value of the Rename File Extension parameter is left blank, the source file is deleted after it is processed. **IMPORTANT**: If you change the default, use only characters that are valid in filenames on your platform. Invalid characters cause the rename to fail and the driver to reprocess the same file repeatedly.
Polling Rate (in Seconds)	When the Publisher channel has finished processing all source files, it waits the number of seconds specified in this parameter before checking for new source files to process.

A.1.6 ECMAScript (Designer Only)

Enables you to add ECMAScript resource files. The resources extend the driver's functionality when Identity Manager starts the driver.

A.2 Global Configuration Values

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

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

In iManager:

1. In iManager, click  to display the Identity Manager Administration page.
2. Open the driver set that contains the driver whose properties you want to edit. To do so:
   - **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 Delimited Text 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.
5. Click *Global Config Values* to display the GCV page.

In Designer:

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

Table A-1 *Global Configuration Values*

Option	Description
Application accepts passwords from Identity Manager	If True, allows passwords to flow from the Identity Manager data store to the connected system.
Identity Manager accepts passwords from application	If True allows passwords to flow from the connected system to Identity Manager.
Publish passwords to NDS password	Use the password from the connected system to set the non-reversible NDS® password in eDirectory.
Publish passwords to Distribution Password	Use the password from the connected system to set the NMAS™ Distribution Password used for Identity Manager password synchronization.
Require password policy validation before publishing passwords	If True, applies NMAS password policies during publish password operations. The password is not written to the data store if it does not comply.
Reset user's external system password to the Identity Manager password on failure	If True, on a publish Distribution Password failure, attempt to reset the password in the connected system by using the Distribution Password from the Identity Manager data store.

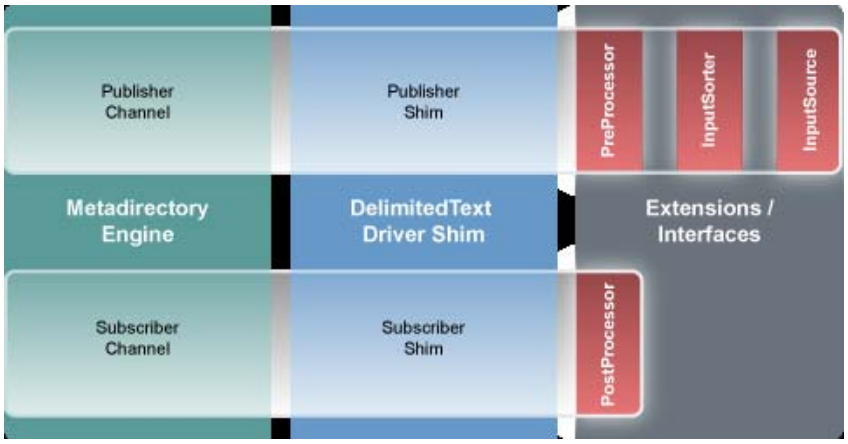
Option	Description
Notify the user of password synchronization failure via e-mail	If True, notify the user by e-mail of any password synchronization failures.
Connected System or Driver Name	The name of the connected system, application, or Identity Manager driver. This value is used by the e-mail notification templates.

Delimited Text Driver Extensions

B

The Delimited Text driver defines different interfaces that you can implement in Java* classes to extend the base functionality of the driver.

Figure B-1 *Java Class Interfaces*



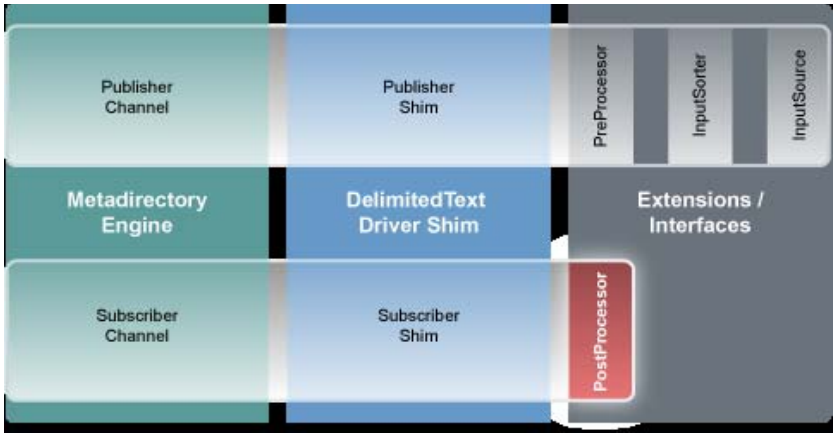
This section includes the following information about using the ImageFile and ImageSource extensions:

- Section B.1, “Using the ImageFile InputSource Extension,” on page 49
- Section B.2, “Customizing ImageFile InputSource,” on page 51
- Section B.3, “Using the ImageFile PostProcessor,” on page 52
- Section B.4, “Customizing the ImageFile Extension,” on page 54

B.1 Using the ImageFile InputSource Extension

The ImageFile InputSource extension allows you to easily import images of type GIF, JPG, and PNG into eDirectory™. The InputSource reads the image files from a specified directory, transforms them to a Base64-encoded string, and passes this string to the driver shim as if it were a normal delimited text string, together with additional information about the image, such as file size, file name, and modification date.

Figure B-2 *InputSource Extension*



Standard rules and style sheets, as used for other implementations of the Delimited Text driver, can be used to further process the image data. The image itself is meant to be stored in an attribute of syntax Octet String or Stream.

- Section B.1.1, "Installing the ImageFile InputSource Extension," on page 50
- Section B.1.2, "Configuring the Driver for the ImageFile InputSource Extension," on page 50

B.1.1 Installing the ImageFile InputSource Extension

The ImageFile InputSource extensions are installed when you install Identity Manager and select the Delimited Text driver. The extensions are configured in the driver parameters. Each extension takes a parameter string that is specified in the driver parameters and passes it to the extension during initialization. Extensions define their own format for the parameter string.

The extensions are included in the `DelimitedTextUtil.jar` file.

B.1.2 Configuring the Driver for the ImageFile InputSource Extension

Use the following table to customize your driver for the ImageFile extension.

Table B-1 *Delimited Text Driver Parameters*

Driver Parameter	XML Name	Sample Values	Purpose
Field Delimiter	field-delimiter	#	Indicates the character that is used to delineate field values in the input files. It must be one character. This must be set to the same character as the delim extension parameter or, if delim is not specified, to #.

Driver Parameter	XML Name	Sample Values	Purpose
Field Names (Field1, Field2, Field3…)	field-names	idx name prefix suffix size modified pic64	A comma-separated list of attribute names that can be referred to in the Schema Mapping rule. In the input files, the fields of the records must correspond to the order and positioning of the names in this list. For example, if you list eight field names in this parameter, then each record of the input files should have eight fields separated by the field delimiter character. The extension defines which fields it supports. The fields as listed here are the ones delivered by the extension. Altering the fields can cause malfunction.
InputSource Class	input-source	com.novell.nds.dirxml.driver.delimitedtext.imagefile	Class name of the input source.
InputSource init string	input-source-params	srcdir=c:\temp\dirxml;renameto=bak;delblacklist=false;renblacklistto=ignore;minsize=1000;debug=true	InputSource initialization parameters. For more information, see Section B.3.2, “Configuring the Driver for the ImageFile Extension,” on page 53.

B.2 Customizing ImageFile InputSource

You can fine-tune the behavior of the ImageFile InputSource by setting the parameters that are discussed in this section. Parameters are passed to the InputSource as a string.

The string must be in the following format:

```
<name1>=<value1>;<name2>=<value2>;<name n>=<value n>
```

Sample:

```
srcdir=c:\temp\dirxml;renameto=bak;delblacklist=false;renblacklist
to=ignore;minsize=1000;debug=true
```

Use the following values to configure the ImageFile properties:

Table B-2 *ImageFile InputSource Parameters*

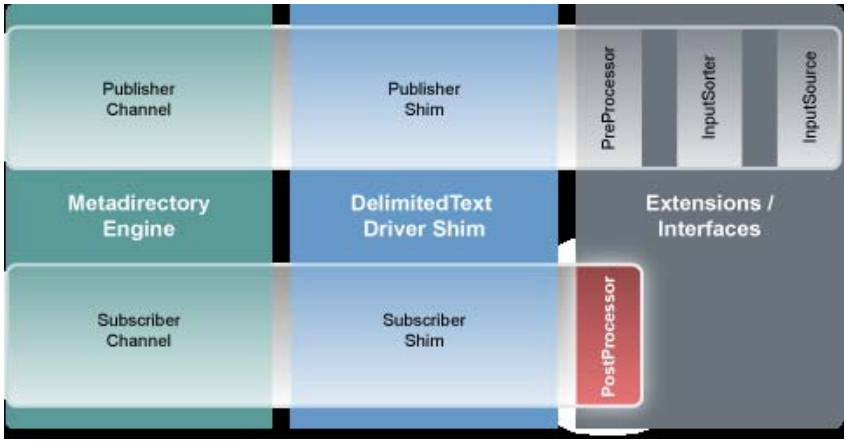
Parameter	Required	Default	Purpose
srcdir	yes		Directory where the image files are stored.
minsize	no	-1	Minimum size of the file to be processed. Number is the number of bytes. Set the value to -1 to disable the filter.

Parameter	Required	Default	Purpose
maxsize	no	-1	Maximum size of the file to be processed, in bytes. Set the value to -1 to disable the filter.
minwidth	no	-1	Minimum width of the image to be processed (number of pixels). Set the value to -1 to disable the filter.
maxwidth	no	-1	Maximum width of the image to be processed (number of pixels). Set the value to -1 to disable the filter.
minheight	no	-1	Minimum height of the image to be processed (number of pixels). Set the value to -1 to disable the filter.
delim	no	#	Delimiter to be used in the created input files. The same delimiter must be configured in the driver parameters for the Delimited Text driver.
consolidate	no	true	Set consolidate to true to produce only one input file for all images. Set consolidate to false to produce an input file for each image.
renameto	no		Specify the suffix you want to be appended to the image files after processing. By default, the renameto parameter is not set and the files are deleted after processing.
debug	no	false	Set to true to turn on debug messages. Debugging is turned off by default.
delblacklist	no	true	Set to false to prevent deletion of the blacklist. The blacklist consists of all the image files that have been filtered out. By default, the blacklist is deleted.
renblacklistto	no		Specify any file suffix you want blacklist image files to be renamed to. By default the renameto parameter is not set and the processed files are handled according to the delblacklist parameter.

B.3 Using the ImageFile PostProcessor

The ImageFile PostProcessor extension allows you to easily export images from the directory to the file system as JPG or PNG files. The PostProcessor further processes the output file generated by the driver. It expects the output file to be in a certain format. The images are contained in the output file as Base64-encoded strings and are then converted into binary blobs and written to files in a specified directory.

Figure B-3 *ImageFile PostProcessor Extension*



Policies and style sheets can be used to control the export process. The images are usually stored in an attribute of syntax Octet String or Stream for binary data.

B.3.1 Installing the ImageFile PostProcessor Extension

The ImageFile PostProcessor extensions are installed when you install Identity Manager and select the Delimited Text driver. The extensions are configured in the driver parameters. Each extension takes a parameter string that is specified in the driver parameters and passes it to the extension during initialization. Extensions define their own format for the parameter string.

The extensions are included in the `DelimitedTextUtil.jar` file.

B.3.2 Configuring the Driver for the ImageFile Extension

- "Driver Module" on page 53
- "Driver Parameters" on page 53

Driver Module

The ImageFile InputSource requires the Delimited Text driver. Select *Java* and enter `com.novell.nds.dirxml.driver.delimitedtext.DelimitedTextDriver` as the driver module. If you want to run the driver with the Remote Loader, select *Connect to Remote Loader* and enter `com.novell.nds.dirxml.driver.delimitedtext.DelimitedTextDriver` as the the driver module in the Remote Loader console.

Driver Parameters

Use the following table to configure the driver. For additional information regarding other driver parameters, refer to the Identity Manager Driver for Delimited Text documentation. (http://www.novell.com/documentation/idm35drivers/delimited/data/bktitle.html)

Table B-3 *ImageFile PostProcessor Parameters*

Parameter	XML Name	Sample Value	Purpose
Field Delimiter	field-delimiter	#	Indicates the character that is used to delineate field values in the input files. It must be one character. This must be set to the same character as the delim extension parameter or, if delim is not specified, to #.
Field Names (Field1, Field2, Field3…)	field-names	idx name prefix suffix size modified pic64	A comma-separated list of attribute names that can be referred to in the Schema Mapping rule. In the input files, the fields of the records must correspond to the order and positioning of the names in this list. For example, if you list eight field names in this parameter, then each record of the input files should have eight fields separated by the field delimiter character. The extension defines which fields it supports. The fields as listed here are the ones delivered by the extension. Altering the fields can cause malfunction.
PostProcessor Class	post-processor	com.novell.nds.dirxml.driver.delimitedtext.imagefile	Class name of the input source.
PostProcessor init string	post-processor-params	destdir=/var/novell/idm/users/output/images;format=png;debug=true	PostProcessor initialization parameters. For more information, see Section B.1.1, “Installing the ImageFile InputSource Extension,” on page 50.

B.4 Customizing the ImageFile Extension

You can fine-tune the behavior of the ImageFile extension by setting parameters. Parameters are passed to the PostProcessor as a string.

The string must be in the following format:

```
<name1>=<value1>;<name2>=<value2>;<name n>=<value n>
```

Sample:

```
destdir=/var/novell/idm/users/output/images;format=png;debug=true
```

Use the following table to customize the ImageFile extension.

Table B-4 *ImageFile Custom Parameters*

Parameter	Required	Default Value	Purpose
destdir	yes		Directory where the image files are stored.
delim	no	#	Used when parsing the output files. The same delimiter should be configured in the Delimited Text driver parameters.
format	no	png	Image format.
debug	no	false	Set the value to true to turn on debugging. Debugging is off by default.