

Novell Identity Manager Driver for Avaya* PBX

3.5.1

www.novell.com

IMPLEMENTATION GUIDE

September 28, 2007



Novell®

Legal Notices

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

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

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

Copyright © 2000-2007 Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher.

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 <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 www.novell.com/documentation.

Novell Trademarks

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

Third-Party Materials

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

Contents

About This Guide	9
1 Overview	11
1.1 Changes in Terminology	11
1.2 The Role of the Identity Manager Driver for Avaya PBX	11
1.3 What's Different about the Avaya Driver?	12
1.4 Benefits of the Avaya Driver	13
1.5 Basic Functionality of the Avaya Driver	14
1.5.1 New Objects Used by the Driver	14
1.5.2 Overview of Driver Functionality	17
1.5.3 Assumptions about the PBX	18
1.5.4 What the Driver Does in the PBX	19
1.5.5 What Is Emulation Mode and Why Should I Use it?	21
1.5.6 Support for Manual Changes in the PBX	21
1.6 Key Driver Features	22
1.6.1 Local Platforms	22
1.6.2 Remote Platforms	23
1.6.3 Role-Based Entitlements	23
1.7 Identity Manager Features	23
2 Planning	25
2.1 Planning Issues for All Configurations	25
2.2 Planning for User Provisioning (Workforce Tree) Implementations	27
2.3 Planning Driver and Replica Placement on Your Servers	28
3 Installation	29
3.1 Prerequisites	29
3.2 Preparing to Install	29
3.2.1 Creating Containers	29
3.2.2 Use Emulation	30
3.2.3 Installing the Driver Preparation	30
3.3 Installing the Avaya Driver During Identity Manager Installation	30
3.4 Importing the Driver Configuration in iManager	31
3.5 Installing the Avaya Driver through Designer	34
3.6 Activating the Driver	36
4 Upgrading the Driver	37
4.1 Changes in Policy Architecture	37
4.2 Upgrading the Driver in Designer	37
4.3 Upgrading the Driver in iManager	40
5 Base Configuration	41
5.1 How the Base Configuration Works	41
5.1.1 How the Subscriber Channel Is Configured	44

5.1.2	How the Publisher Channel Is Configured	45
5.2	Planning for the Base Configuration	46
5.3	Setting Up the Base Configuration	46
6	Activating the Driver	47
7	Workforce Tree Configuration	49
7.1	How the Workforce Tree Configuration Works	49
7.1.1	How the Driver Uses the ObjectID to Identify and Update Users.	51
7.1.2	How the Subscriber Channel Is Configured	52
7.1.3	How the Publisher Channel Is Configured	54
7.1.4	User Events That Can Trigger a Work Order.	55
7.2	Planning for the Workforce Tree Configuration	56
7.3	Setting Up the Workforce Tree Configuration.	56
7.4	Using Log/Reporting Functionality to Report Warnings	56
8	Work Order Database Configuration	57
8.1	Configuring a Work Order Database Configuration	57
8.1.1	How To Configure the Subscriber Channel	60
8.1.2	How To Configure the Publisher Channel	61
8.2	About Work Order Systems	61
8.3	Planning for the Work Order Database Configuration	61
8.4	Setting Up the Work Order Database Configuration	61
8.5	Using the Log/Reporting Functionality to Report Warnings	61
9	Managing the Driver	63
9.1	Starting, Stopping, or Restarting the Driver	63
9.1.1	Starting the Driver in Designer	63
9.1.2	Starting the Driver in iManager	63
9.1.3	Stopping the Driver in Designer	63
9.1.4	Stopping the Driver in iManager.	63
9.1.5	Restarting the Driver in Designer	64
9.1.6	Restarting the Driver in iManager	64
9.2	Migrating and Resynchronizing Data	64
9.3	Using the DirXML Command Line Utility	64
9.4	Viewing Driver Versioning Information	65
9.4.1	Viewing a Hierarchical Display of Versioning Information	65
9.4.2	Viewing the Versioning Information As a Text File	66
9.4.3	Saving Versioning Information	68
9.5	Reassociating a Driver Set Object with a Server Object	69
9.6	Changing the Driver Configuration	69
9.7	Storing Driver Passwords Securely with Named Passwords	70
9.7.1	Using Designer to Configure Named Passwords	70
9.7.2	Using iManager to Configure Named Passwords	71
9.7.3	Using Named Passwords in Driver Policies	72
9.7.4	Using the DirXML Command Line Utility to Configure Named Passwords	73
9.8	Adding a Driver Heartbeat	76
10	Managing the Identity Manager Driver for Avaya PBX	79
10.1	Changing How Often the Driver Performs Work Orders.	79

10.2	Changing the Location of PBX Site, Work Orders, User, and Extension Objects	79
10.3	Managing Existing Users	80
11	Synchronizing Objects	83
11.1	What Is Synchronization?	83
11.2	When Is Synchronization Done?	83
11.3	How Does the Metadirectory Engine Decide Which Object to Synchronize?	84
11.4	How Does Synchronization Work?	85
11.4.1	Scenario One	85
11.4.2	Scenario Two	87
11.4.3	Scenario Three	88
12	Backing Up the Driver	89
12.1	Exporting the Driver in Designer	89
12.2	Exporting the Driver in iManager	89
13	Security: Best Practices	91
14	Troubleshooting the Driver	93
14.1	Troubleshooting Driver Processes	93
14.1.1	Viewing Driver Processes	93
A	Schema for PBX Management	101
A.1	pbxSite Object	101
A.2	DirXML-nwoWorkOrder Object	103
A.3	DirXML-pbxExtension Object	108
A.4	DirXML-pbxAudixSubscriber Object	110
A.5	User Objects and Their Identity Manager Associations	111
B	DirXML Command Line Utility	113
B.1	Interactive Mode	113
B.2	Command Line Mode	122
C	Properties of the Driver	127
C.1	Driver Configuration	127
C.1.1	Driver Module	128
C.1.2	Driver Object Password	128
C.1.3	Authentication	129
C.1.4	Startup Option	130
C.1.5	Driver Parameters	131
C.2	Global Configuration Values	132
C.3	Named Passwords	132
C.4	Engine Control Values	133
C.5	Log Level	135
C.6	Driver Image	136
C.7	Security Equals	136
C.8	Filter	136

C.9	Edit Filter XML	137
C.10	Misc	137
C.11	Excluded Users	138
C.12	Driver Manifest	138
C.13	Driver Inspector	139
C.14	Driver Cache Inspector	139
C.15	Inspector	140
C.16	Server Variables	140

About This Guide

The Identity Manager 3.5.1 Driver for Avaya* PBX lets you use Identity Manager to centrally manage PBX extensions and work orders and keep user phone numbers up-to-date. Newer driver features allow you to also configure voice mail through Audix. This guide explains what the Avaya driver does and how the driver can be customized through three sample configurations that are provided for instructional purposes.

This guide contains the following sections:

- ◆ Chapter 1, “Overview,” on page 11
- ◆ Chapter 2, “Planning,” on page 25
- ◆ Chapter 3, “Installation,” on page 29
- ◆ Chapter 4, “Upgrading the Driver,” on page 37
- ◆ Chapter 5, “Base Configuration,” on page 41
- ◆ Chapter 6, “Activating the Driver,” on page 47
- ◆ Chapter 7, “Workforce Tree Configuration,” on page 49
- ◆ Chapter 8, “Work Order Database Configuration,” on page 57
- ◆ Chapter 9, “Managing the Driver,” on page 63
- ◆ Chapter 10, “Managing the Identity Manager Driver for Avaya PBX,” on page 79
- ◆ Chapter 11, “Synchronizing Objects,” on page 83
- ◆ Chapter 12, “Backing Up the Driver,” on page 89
- ◆ Chapter 13, “Security: Best Practices,” on page 91
- ◆ Chapter 14, “Troubleshooting the Driver,” on page 93
- ◆ Appendix A, “Schema for PBX Management,” on page 101
- ◆ Appendix B, “DirXML Command Line Utility,” on page 113
- ◆ Appendix C, “Properties of the Driver,” on page 127

Audience

This manual is for Novell® Identity Manager administrators, Avaya PBX communications system administrators, and others who manage user accounts in eDirectory™, PBX phone extensions, and PBX work orders.

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation, or go to [Novell's Feedback Web site \(http://www.novell.com/documentation/feedback.html\)](http://www.novell.com/documentation/feedback.html) and enter your comments there.

Documentation Updates

For the most recent version of this document, see the [Drivers Documentation Web site \(http://www.novell.com/documentation/idm35drivers/index.html\)](http://www.novell.com/documentation/idm35drivers/index.html).

Additional Documentation

For documentation on using Novell Identity Manager and the other drivers, see the [Identity Manager Documentation Web site \(http://www.novell.com/documentation/idm35/\)](http://www.novell.com/documentation/idm35/).

Documentation Conventions

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

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

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

The Identity Manager Driver for Avaya PBX lets you use the Identity Vault to centrally manage PBX extensions and work orders and keep user phone numbers up-to-date.

This configurable solution gives you the ability to automate work order and PBX processes, such as provisioning a phone extension for new users; moving, modifying or disabling existing extensions; disconnecting extensions, updating phone number data for User objects in the Identity Vault, and provisioning voice mail/Audix* for phone extensions.

This product can be an important part of a provisioning solution.

In this section:

- ♦ [Section 1.1, “Changes in Terminology,” on page 11](#)
- ♦ [Section 1.2, “The Role of the Identity Manager Driver for Avaya PBX,” on page 11](#)
- ♦ [Section 1.3, “What’s Different about the Avaya Driver?,” on page 12](#)
- ♦ [Section 1.4, “Benefits of the Avaya Driver,” on page 13](#)
- ♦ [Section 1.5, “Basic Functionality of the Avaya Driver,” on page 14](#)
- ♦ [Section 1.6, “Key Driver Features,” on page 22](#)
- ♦ [Section 1.7, “Identity Manager Features,” on page 23](#)

1.1 Changes in Terminology

The following terms have changed from earlier releases:

Table 1-1 *Changes in Terminology*

Earlier Terms	New Terms
DirXML®	Identity Manager
DirXML Server	Metadirectory server
DirXML engine	Metadirectory engine
eDirectory™	Identity Vault (except when referring to eDirectory attributes or classes)

1.2 The Role of the Identity Manager Driver for Avaya PBX

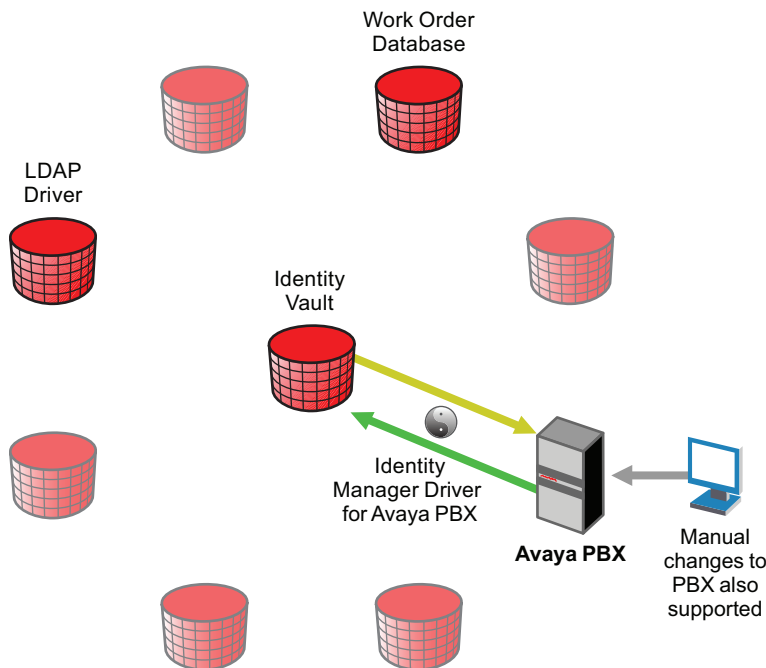
The Identity Manager Driver for Avaya PBX communicates with the PBX and carries out work orders by logging in just as a PBX administrator would. The driver uses information from the Identity Vault as well as customizable settings and policies to configure the extension correctly. After configuration is completed, the driver publishes the new phone number to the Identity Vault, and the phone can be delivered and physically installed.

The driver can be used in conjunction with other systems such as a human resources application or work order database to provide additional functionality.

The following diagram shows some of the systems you might have in your environment, and shows that you can connect the PBX to the Identity Vault using the Identity Manager Driver for Avaya PBX.

The terminal shown in the diagram demonstrates that you can still access the PBX and make changes manually, if necessary. However, with the driver in place, you shouldn't need to make changes directly in the PBX very often. The purpose of the driver is to allow you to use work orders to control changes in the PBX, so you don't need to make them manually. Work orders can be triggered by events elsewhere in your environment, depending on what other systems you have connected using Identity Manager.

Figure 1-1 The Avaya PBX Driver Connecting the Identity Vault To One or More PBX Sites



1.3 What's Different about the Avaya Driver?

The Identity Manager Driver for Avaya PBX is different from some other drivers in the following ways:

- ♦ Through policies, the driver can map users to phone extensions, rather than users to users.

Some drivers are used for user account administration, mapping user accounts in one system to user accounts in another. The Avaya PBX driver can help you maintain correct phone information for user objects, but the PBX understands only extensions and does not contain the concept of a user account, so the user objects in Identity Vault must be connected to extensions in the PBX by using the Object ID in the work order. See [Section 10.3, "Managing Existing Users," on page 80](#).

- ♦ The ObjectID in the work order (such as user object entry ID, or employee ID) is the piece of information that allows a user object to be updated with new phone information after a task is

performed in the PBX. This ID should be a unique way to identify the user, and a policy must be customized to match the attribute that you use.

The driver places the Object ID in the Cable field of the extension in the PBX. This field has a character limit of 5 characters.

Even if the user has an Identity Manager association for this driver through a policy, in most cases the user cannot be updated unless the ObjectID is entered correctly. In fact, a user object can only be updated if the ObjectID is correct, even if the user does not yet have an Identity Manager association for this driver. See [Section 10.3, “Managing Existing Users,” on page 80](#).

- ◆ The Publisher performs tasks in the Avaya PBX, rather than the Subscriber.

For many drivers, the Subscriber performs changes in the third-party application in response to events in eDirectory. However for this driver, the Publisher is the agent that performs work orders in the PBX. The Subscriber merely picks up events such as Add User events, creates work orders if configured to do so, and sends work orders to the Publisher if they are marked Send to Publisher or DoItNow.

There are several reasons for this design; for example, the Publisher has the ability to run at a certain time of day, which is desirable for allowing you to specify that work orders be performed after business hours.

- ◆ Creating a robust test environment can be a challenge (often, the only PBX available in an environment is the production PBX). If this is the case, you can use emulation mode for testing. One option is to set aside a certain range of extensions on the production PBX to use for testing, and use extra caution because it’s not an entirely separate test environment.
- ◆ You can manage existing users without using the `Migrate into NDS` command. See [Section 10.3, “Managing Existing Users,” on page 80](#).

1.4 Benefits of the Avaya Driver

Benefits of using the driver include the following:

- ◆ Eliminates redundant data entry.

If you use a work order system without the driver, you must enter the data twice, once in the work order and again in the PBX when performing the work order. With the driver, you enter the data once in the work order, and then the driver performs it for you, reducing the possibility of human error.

In fact, in a provisioning implementation you can avoid human intervention entirely for actions such as new hires, moves, and deletions; it can all be based on changes in a human resources application.

- ◆ Allows automated provisioning of extensions for new users.
- ◆ Allows you to use the Identity Vault as your work order database for PBX tasks, if you don’t already have one.

An iManager interface lets you create and track PBX work orders.

- ◆ Helps reduce costs and data entry errors by letting you enforce business policies for phone usage.

For example, you could ensure that only users in the Localization, Sales, and Human Resources departments can make international calls. Another example could be making sure that extensions for call center employees are always non-DID extensions.

- ◆ Allows updates to be sent to the Identity Vault when an extension is changed directly in the PBX.

This means that after deploying the driver you can still make changes in the PBX manually when necessary.

When the driver prepares to perform work orders at the specified time, it queries the PBX to see if changes have been made. If a manual change has been made to a PBX extension, the driver can detect the change. If the PBX administrator who made the manual change also enters the user ID in the PBX to identify the user of the extension, then the driver can update the user object in eDirectory.

- ◆ Audix support to the Avaya driver. Through this support, the driver can add voice mail to an extension, remove voice mail from an extension, or modify the attributes of the voice mail of an extension.

For Audix support, the driver uses the following work order commands: add, change and remove.


1.5 Basic Functionality of the Avaya Driver

- ◆ [Section 1.5.1, “New Objects Used by the Driver,” on page 14](#)
- ◆ [Section 1.5.2, “Overview of Driver Functionality,” on page 17](#)
- ◆ [Section 1.5.3, “Assumptions about the PBX,” on page 18](#)
- ◆ [Section 1.5.4, “What the Driver Does in the PBX,” on page 19](#)
- ◆ [Section 1.5.5, “What Is Emulation Mode and Why Should I Use it?,” on page 21](#)
- ◆ [Section 1.5.6, “Support for Manual Changes in the PBX,” on page 21](#)

1.5.1 New Objects Used by the Driver

Using four new object classes in eDirectory, the Identity Manager Driver for Avaya PBX performs work orders, records the results, and provides extension information. The Avaya driver also supports Audix account provisioning. For a description of the schema for these objects, see [Appendix A, “Schema for PBX Management,” on page 101](#).

DirXML-pbxSite

 DirXML-pbxSite represents the PBX.

The driver depends on the information in PBX Site objects to know which sites to manage, and how to connect to them.

An iManager plug-in is provided to help you set up these sites. In iManager, select *PBX Utilities > Site Management*. The following figure shows an example of the interface for setting up a PBX site.

Figure 1-2 The PBX Site Page

Novell iManager - Microsoft Internet Explorer

File Edit View Favorites Tools Help

PBX Site: LabPBX

PBX Site

Configure this PBX site object to enable the driver to connect to the PBX. * Required

Access Type:

Are jacks hot?

General Information

PBX name: *	pbx name
Login name: *	login name
Password: *	*****
Extension length: *	5
Beginning number of DID extension range: *	12345
Ending number of DID extension range: *	23456
Beginning number of non-DID extension range:	23457
Ending number of non-DID extension range:	3000

OK Cancel Apply

nwoWorkOrder

DirXML-nwoWorkOrder represents work orders.

This object lets you indicate what actions you want the driver to perform in the PBX, and specify other details such as when the work order should be performed, the object ID and display name of the person who owns the extension, and whether a specific extension number is requested. After the driver performs the work order, it updates the work order with information such as the status (Configured, Error, etc.).

An iManager plug-in is provided to help you create and maintain work orders. In iManager, select *PBX Utilities > Work Order Management*. The following figure shows an example of the interface for creating a work order.

Figure 1-3 The Work Order Page

Novell iManager - Microsoft Internet Explorer

Work Order: status pending

PBX

PBX Work Order

The work order object is used to tell the driver what tasks to perform in the PBX and to allow the driver to record the results.

Action: Install

* Required

PBX name: *

Description:

Extension:

Extension type: <Select an extension type>

Do it now: false

Due date: * 5/3/2004

Node:

Phone type:

Status: Pending

Work order number:

Send to publisher: false

Duplicate extension:

Room:

Display name:

Object ID:

Port:


OK Cancel Apply

pbxExtension

 DirXML-pbxExtension represents extensions.

After performing a work order, the driver shim sends one of these objects to represent the work that was done. For example, if a new extension was configured, a new extension object is sent with the correct extension number and display name. The driver updates the work order with information such as the status (Actions taken if successful, Configured, Error, etc.) If you create a new AudixSubscriber object, a new DirXML-pbxAudixSubscriber object is sent to the engine.

pbxAudixSubscriber

 DirXML-pbxAudixSubscriber represents Audix support.

With Audix support for the Avaya driver, the driver can add voice mail to an extension, remove voice mail from an extension, or modify the attributes of the voice mail of an extension. However, this driver support does not replace general administration of the Avaya Audix system.

Adding Audix support to the current Avaya driver requires the driver to access and manipulate PBX subscriber objects. This is performed by extending the eDirectory schema to include a DirXML-pbxAudixSubscriber object and by adding a new site object that defines the Audix server.

1.5.2 Overview of Driver Functionality

When configuring the driver, you have the option to run the driver in emulation mode. Although this mode of driver operation is optional, we highly recommend its use. Emulation mode enables you to fully test the driver and its policies before connecting to a live PBX system. When you run in emulation mode, the driver emulates the PBX specified by the `pbxSite` object. All driver actions are directed to the LDAP site that you specify in the Configuration Parameters. To run in emulation mode, locate the `pbxSite` object and set the `DirXML-AccessType` attribute to “emulate.” This causes the driver to emulate the PBX. For more information, see [“What Is Emulation Mode and Why Should I Use it?” on page 21](#).

The driver shim itself functions in the following basic way:

1. When the driver starts, it reads the information for the `DirXML-pbxSite` objects from the container you specified in the driver parameters.

These objects tell the driver which PBXs to perform work orders on, and they provide other details such as which number ranges can be used for extensions.

2. At the polling time or interval you set, the driver shim “wakes up,” and the Publisher queries the PBX for all the extensions. If it detects that a change has been made manually since the last polling interval, it sends a `DirXML-pbxExtension` object to the Identity Vault representing the change.

See [“Support for Manual Changes in the PBX” on page 21](#).

3. The Publisher then polls for `DirXML-nwoWorkOrder` objects in eDirectory in the container you specified in the driver parameters. It checks the `DirXML-nwoDueDate` and `DirXML-nwoStatus` attributes on the work orders to see which of them need to be performed.
4. The Publisher performs the work orders that are due, completing the appropriate action based on attributes of the `DirXML-nwoWorkOrder` objects. If a value was present in the work order for the `DirXML-nwoObjectID` attribute, the Object ID is placed in the Cable field for that extension in the PBX.
5. The Publisher updates the `DirXML-nwoWorkOrder` with the results.

For example, if a work order to install a new extension is successful, the `DirXML-nwoStatus` attribute is updated with the value “Configured.” If no extension number was requested in the work order, or if the requested extension was not available, the Publisher specifies which extension was chosen.

6. For installation work orders, the Publisher also sends a `DirXML-pbxExtension` object to the Identity Vault for each work order, representing the results.

For example, if a work order to install a new extension is successful, a `DirXML-pbxExtension` object is sent to the Identity Vault with the new extension in the `DirXML-nwoExtension` attribute.

7. For add Audix subscriber work orders, the Publisher also sends a new `DirXML-pbxAudixSubscriber` object to the Identity Vault.

The driver can also perform a work order immediately instead of waiting for the next polling interval, if you indicate Do It Now in the work order. Work orders with this attribute are immediately processed by the Publisher channel.

You can customize the driver to enhance what it does. The sample configurations demonstrate some of these customizations.

For example, you can use driver policies to do the following:

- ◆ Create a work order when a new user object is added to the Identity Vault in order to automate an extension assignment to a new employee. This could be further automated by using an additional Identity Manager driver to automatically create user objects in the Identity Vault when they are created in a human resources application.
- ◆ Assign certain phone restrictions based on the location or job title indicated for the user object in the Identity Vault. For example, you could configure the driver to use non-DID extensions for employees in the call center.
- ◆ Using an additional Identity Manager driver, cause work order objects from another application to be synchronized to the Identity Vault and performed by the Avaya PBX driver.
- ◆ Transform an add DirXML-pbxExtension object coming from the driver into an add and a modification of a User object, so you can update the User object with the new extension when a work order completes.
- ◆ Add other customizations to fit your business processes.

To demonstrate some of the things you can do with the Avaya PBX driver, two sample driver configurations are provided. A third kind of configuration is also described in this guide, but the sample scripting provided is not a complete driver configuration.

These three kinds of configurations are discussed in the following chapters:

- ◆ [Chapter 5, “Base Configuration,” on page 41](#)
- ◆ [Chapter 7, “Workforce Tree Configuration,” on page 49](#)
- ◆ [Chapter 8, “Work Order Database Configuration,” on page 57](#)

1.5.3 Assumptions about the PBX

The PBX is the Public Branch Avaya PBX or corporate phone system. The PBX can be centralized or distributed across buildings or sites. A PBX system can be composed of several PBX cabinets, including remote cabinets, controlled by a master PBX cabinet. Typically, each phone is physically cross-connected to a PBX digital port.

Because the driver is designed to be generic, the PBX can be used in any environment, from an enterprise call center system to a small office key system.

The Identity Manager Driver for Avaya PBX relies on the following assumptions:

1. Responsibility for PBX administration lies with the user rather than a third party.
2. The PBX administrator has sufficient training and certification, and appropriate access rights to the PBX.
3. The PBX supports administration by telnet.
4. Voice jacks are either “hot jacks” or “cold jacks.” The driver supports both kinds of environments. See [“Voice Jacks” on page 19](#).
 - ◆ [“Voice Jacks” on page 19](#)
 - ◆ [“Querying the PBX” on page 19](#)

Voice Jacks

The behavior of the driver depends on whether or not the PBX environment has hot jacks (jacks that are permanently cross-connected to live PBX ports) or cold jacks (jacks that are cross-connected at the time of phone installation).

Hot Jacks

Hot jacks are simpler from a configuration standpoint, if the PBX supports port selection from the phone set. If the PBX does not support port selection from the phone set, the cold jack process is followed. (See [“Cold Jacks” on page 19.](#)) The driver assigns the port specified in the work order. If no port is specified, the driver will “X out” the port.

When a work order requests a new phone, the driver creates the extension in the PBX, but leaves the port unassigned. If the work order did not request a specific extension number, the driver assigns the first available number in numeric order. When the phone is delivered and plugged in, punching in a code activates the extension on that phone by automatically assigning the port (if the port was “Xed out.”)

When moving an existing extension, the driver simply unassigns the port, meaning it is Xed out. This deactivates the extension. When the phone is delivered and plugged in at the new location, punching in the activation code assigns the new port to the extension.

Cold Jacks

If the PBX environment has cold jacks, the installation process requires you to specify which node the user is in as part of the work order. If the extension is not given, the next available number is assigned. The driver also scans for available ports that work with the phone type and are in the correct node. The new port is assigned to the extension and is noted in the work order.

When the phone is delivered, a technician must cross-connect the port to the new jack.

If you have cold jacks, but you prefer that the driver X out the port instead of automatically choosing one, you can specify the hot jacks option. When the phone is physically installed, the driver sees the change after its next polling cycle and changes the status of the work order.

Querying the PBX

Because of the limitations of the PBX system, the driver queries the PBX only for DirXML-pbxExtension and DirXML-pbxAudixSubscriber extensions.

The PBX doesn’t support queries on the display name or the field containing a user identifier. To find those pieces of data when querying the PBX, you need to create a custom policy configuration that queries the PBX for all the extensions and their associated data, and then searches through the entire list to find the value you want.

1.5.4 What the Driver Does in the PBX

At the most basic level, the driver can perform several actions in the PBX: install, disable, enable, move, modify, setcor, disconnect. The following list describes the main tasks you can perform using work orders.

- ◆ **Install:** Assign an extension in the PBX at a given location, and activate the extension.

You can request a specific extension number, and the driver assigns that number if it's available in the PBX. If it's not available, or if no extension is specified in the work order, the driver assigns the next available extension number in numeric order within the range of extensions specified in the PBX site object.

- ◆ **Disable:** Stop allowing an extension to be used, but leave it installed in case you want to enable it again in the future.
- ◆ **Enable:** Allow an extension to be used again after it has been disabled.
- ◆ Move an extension in one of the following ways:
 - ◆ Deactivate it in one location and activate it in another location within the same PBX system.
 - ◆ With the use of custom policies, deactivate it in one PBX system and activate it in another PBX system.
- ◆ **Setcor:** Use the setcor command to change restrictions for the extension, such as whether long distance or international calls are allowed.
- ◆ **Disconnect:** Remove an extension from the PBX.
- ◆ Set values such as the following, based on data you provide in the work order:
 - ◆ The date to complete the desired action.
If the DoItNow flag is set, the driver performs the action immediately.
The driver is configured so that at a specified time once a day it performs work orders that are due that day. You can also set a polling interval for performing work orders.
 - ◆ The type of phone (DID or non-DID).
 - ◆ Restrictions for use of the phone extension, such as whether long distance or international calls are allowed.
 - ◆ The port number for the phone connection (necessary if you use cold jacks).
- ◆ Query the PBX to find out what manual changes have been made since the last time the driver polled for work orders. This feature supports manual changes to the PBX extensions. However, the driver does not look for changes to Audix subscribers.

At the specified time of day or polling interval, the Publisher wakes up to perform work orders. Before polling for and performing work orders, the Publisher first queries the PBX to find out whether anything has changed since the last time the Publisher performed work orders. This means that you can make changes manually in the PBX, and the driver is made aware of those changes when it next communicates with the PBX.

This feature makes the driver implementation more flexible, because you are not limited to making changes only through the driver. In addition, if you make sure the object ID is inserted the label field in the PBX whenever you make a manual change, you can use this feature to automate updates to user objects to reflect changes in the PBX made manually, as well as changes made by the driver. For example, in the workforce tree sample configuration, if you install an extension and insert the object ID into the label field, the driver discovers that change and automatically updates the user object with the new extension.

NOTE: The above feature is true for Avaya, but not for Audix.

See [“Support for Manual Changes in the PBX” on page 21](#).

1.5.5 What Is Emulation Mode and Why Should I Use it?

For all new Identity Manager products, we recommend that you configure and use them first in a test environment. When you are comfortable with the test environment, you can move into production. Most companies, however, don't have PBX systems dedicated to testing. To aid you in testing and debugging, the driver has a built-in emulation mode. In emulation mode, you configure the driver as if you were connecting to your PBX system. But instead of connecting to the PBX, the driver is able to add, modify, and delete extension and Audix Subscriber objects in an LDAP container, which is simulating the PBX. You can fully test policies without affecting the live PBX. It enables you to make PBX changes more easily because you can simulate and watch work order processing before moving to a production environment.

How Does Emulation Work?

When the driver loads and detects the emulation settings, it uses the emulation parameters set during configuration. The driver binds to the LDAP directory and looks for a PBX site container that holds extensions. If this container does not exist, the driver creates a container with the same name as the PBX name in the site object. During emulation, this is the container the driver looks for when making changes to extension objects. All read/write activity occurs in the PBX site container, and the driver treats the container like it is a PBX system. If a work order is created to install a new extension, you should look in this container to verify that a DirXML-Extension object was created.

How Do I Configure Emulation Mode?

When configuring the driver's parameters, there are five parameters (in the Publisher settings) that you need to specify in order to configure the driver for emulation. They are:

- ◆ The IP address of the LDAP host you want to use for emulation. This can be the Identity Vault or any other LDAP directory.
- ◆ The LDAP port that the host uses for LDAP.
- ◆ The DN of the login user on the LDAP host you are using.
- ◆ The password of that user.
- ◆ The DN for the container that will hold the emulated extensions.

You also need to change the Site object for the PBX, and change the DirXML-AccessType attribute to `emulate` or `AudixEmulate`. For more information on these settings, see [Table 3-3 on page 33](#).

1.5.6 Support for Manual Changes in the PBX

The preferred method of performing PBX tasks is to create a work order and let the driver configure the PBX. However, manual changes made in the PBX are also supported.

When the driver prepares to perform work orders, at the polling interval or time of day you specify, it first queries the PBX for the extensions to see if anything has changed since the last time the driver communicated with the PBX. This functionality allows the driver to update the Identity Vault to reflect changes that have been made manually, as much as possible.

The following table indicates the changes the driver can detect in the PBX if the changes were made manually, and what the driver can do in response to them.

Table 1-2 Changes the Driver Can Detect in the PBX

Change made manually in the PBX	Can the driver detect the change?	Can the driver update the Identity Vault in response to the change?	How the driver can update the Identity Vault
Install a new extension	Yes	Yes, if the ObjectID is entered correctly in the PBX.	The driver creates a new nwoExtension object.
Modify the display name	Yes	Yes, if the ObjectID is entered correctly in the PBX.	The driver modifies the DisplayName attribute of the nwoExtension object. (In the workforce tree configuration, no change would be necessary.)
Modify the ObjectID	Yes	Yes, if the ObjectID is entered correctly in the PBX.	The driver can update the nwoExtension object with a new ObjectID.
Setcor	No	No update to Identity Vault is necessary.	
Disable an extension	Yes	Yes	The driver updates the nwoExtension object in the Identity Vault.
Disconnect an extension	Yes	No	In the base configuration, the driver can delete the nwoExtension object that represented the extension. You can write a policy to update all users who have this extension in their phone lists, by removing the extension from the list. Instead of relying on the ObjectID in this case, the driver determines which users to update by querying Identity Vault for User objects that have that extension.

1.6 Key Driver Features

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

- ♦ [Section 1.6.1, “Local Platforms,” on page 22](#)
- ♦ [Section 1.6.2, “Remote Platforms,” on page 23](#)
- ♦ [Section 1.6.3, “Role-Based Entitlements,” on page 23](#)

1.6.1 Local Platforms

The Avaya PBX driver can be installed locally on the following platforms:

- ♦ Windows* NT*, 2000, or 2003 with the latest Service Patch
- ♦ Novell® Open Enterprise Server with the latest Support Pack
- ♦ Linux Red Hat* 3.0 and 4.0 for AMD64/EM64T
- ♦ SUSE® LINUX Enterprise Server 9 and 10 (with latest Support Pack)

- ♦ Solaris* 9 or 10
- ♦ AIX* 5.2L, v5.2 and v5.3

1.6.2 Remote Platforms

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

- ♦ Windows NT, 2000, or 2003 with the latest Service Patch
- ♦ Novell Open Enterprise Server with the latest Support Pack
- ♦ Linux Red Hat* 3.0 and 4.0 for AMD64/EM64T
- ♦ SUSE® LINUX Enterprise Server 9 and 10 (with latest Support Pack)
- ♦ Solaris* 9 or 10
- ♦ AIX* 5.2L, v5.2 and v5.3

For more information about installing the Remote Loader services, see “[Installing the Remote Loader](#)” in the *Novell Identity Manager 3.5.1 Administration Guide*.

1.6.3 Role-Based Entitlements

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

1.7 Identity Manager Features

Identity Manager includes new features. For more information, refer to the “[What's New in Identity Manager 3.5.1?](#)” in the *Identity Manager 3.5.1 Installation Guide*.

Planning issues can vary significantly depending on your environment and goals; this planning section provides a starting point for developing your custom implementation.

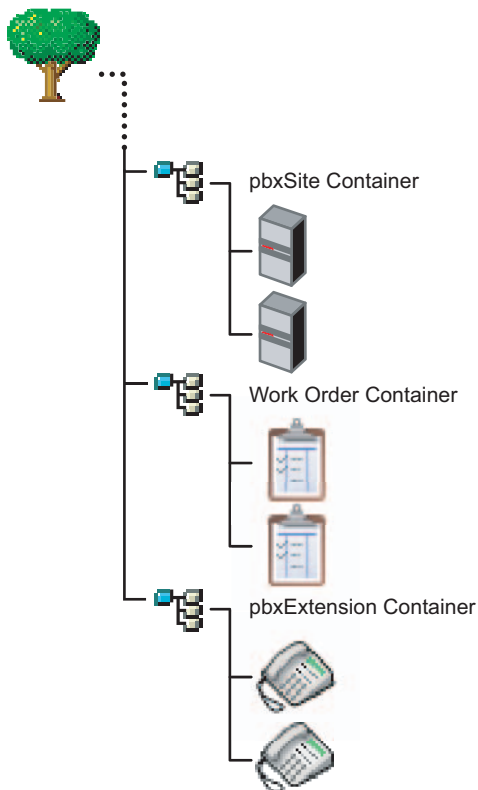
- ♦ [Section 2.1, “Planning Issues for All Configurations,”](#) on page 25
- ♦ [Section 2.2, “Planning for User Provisioning \(Workforce Tree\) Implementations,”](#) on page 27
- ♦ [Section 2.3, “Planning Driver and Replica Placement on Your Servers,”](#) on page 28

2.1 Planning Issues for All Configurations

The items in this section should be relevant regardless of how you want to implement the driver.

- ♦ Identify or create containers to hold the new kinds of objects used by the driver, as shown in the following figure and described in the list of questions after the figure.

Figure 2-1 *Example of Containers for New Objects*



- ♦ Which container do you want to use for pbxSite objects?

You must create a pbxSite object to represent each of your PBXs. The driver queries for these objects at startup so it knows how to communicate with each PBX. For more information, see [Section A.1, “pbxSite Object,”](#) on page 101.

- ♦ Which container do you want to use for nwoWorkOrder objects?

These are the objects used to tell the driver which tasks to perform in the PBX. For more information, see [Section A.2, “DirXML-nwoWorkOrder Object,” on page 103](#).

- ◆ Are you going to use nwoExtension objects, and if so, which container do you want to use for them?

In the base configuration, nwoExtension objects are used to represent extensions, so you can see the results of the tasks the driver performs in the PBX. For more information, see [Section A.3, “DirXML-pbxExtension Object,” on page 108](#).

Depending on your implementation, you might not need to use nwoExtension objects. Instead, you can configure your policies to transform events for nwoExtension objects from the driver into events that update phone information for User objects, as demonstrated in the Workforce Tree configuration (see [Chapter 7, “Workforce Tree Configuration,” on page 49](#)).

- ◆ Gather all the information about your PBX sites that you need to enter into each pbxSite object.

For example, you need to know the answers to questions like the following:

- ◆ How many digits are the extensions?
- ◆ How many nodes do you have?

For a list of all the requirements you need to meet and the information you need to provide in the pbxSite objects, see [Section A.1, “pbxSite Object,” on page 101](#).

- ◆ Do you have hot jacks, or cold jacks?

This aspect of your environment affects what information you must supply when creating a work order. The driver needs more information to perform tasks for cold jacks than it does for hot jacks, so for a cold jack environment more attributes are required on a nwoWorkOrder object. See the list of attributes required for hot jacks and cold jacks in [Section A.2, “DirXML-nwoWorkOrder Object,” on page 103](#).

If you have cold jacks, but you prefer that the driver “X out” a port instead of automatically choosing one, you can specify the hot jacks option.

- ◆ What are the duplicate extensions for each phone type?

You need to reproduce the mapping in the policies when a work order asks the driver to install an extension. The driver refers to the duplicate extension policy on the PBX to create the new extension.

Duplicate (dupe) extensions are set up by the PBX administrator in the PBX, using extensions that are outside the range of extensions that are available for use as DID and non-DID phone numbers. They are used when installing an extension. These dupe extensions are used only as a template to set up each phone type (such as phone type 8410) correctly.

In each work order the duplicate extension for the phone type needs to be specified, so the PBX has the right template to follow. The policies for the sample configurations demonstrate how to map the phone type to the duplicate extension.

- ◆ When do you want the driver to perform work orders?

You can control the timing using polling interval, time of day, or both.

Of course, if the work order is marked DoItNow, the driver performs it immediately and doesn’t wait for a polling interval or time of day.

- ◆ How will you create a test environment for testing the driver with the PBX?

Creating a robust test environment can be a challenge, because often the only PBX available in an environment is the production PBX. To solve this problem, we recommend using the

driver's emulation mode. See [Section 1.5.5, “What Is Emulation Mode and Why Should I Use it?”](#) on page 21 for more information.

2.2 Planning for User Provisioning (Workforce Tree) Implementations

This section explains the additional planning issues you need to consider when creating an implementation of the driver that is like the sample workforce tree configuration provided. (See [Chapter 7, “Workforce Tree Configuration,”](#) on page 49.)

- ♦ What ID will you use to identify the user for whom the work order is being performed?

This ID number is entered in the work order as the ObjectID attribute, and you can use a custom policy to cause Identity Manager to match the work order with the corresponding user.

The sample workforce tree configuration provided uses the eDirectory™ Workforce ID for the User object, but you could use the employee ID, or some other number that is used in your organization. This number must be stored as an attribute of the user object so the driver can find it.

Because this number is stored in the PBX in the Cable field, it can be only 5 digits long.

Keep in mind that this ID must also be entered in the PBX for any extensions entered manually.

- ♦ How will you specify the user's location so that the policies can determine where in the PBX to configure the extensions port?

When you are planning how to determine which PBX to configure the user's extension in, it's important to understand how the PBX sites are configured. A PBX cabinet can be configured as a remote cabinet, meaning that it is controlled by a master PBX but is at a different office or even in a different city. This can mean that a user location in Human Resources might not have a simple mapping to a PBX site and location within the PBX system.

- ♦ How will you determine whether a user should receive a DID or a non-DID extension?

You need to determine what business rules to follow. For example, you could customize the policies to assign non-DID extensions to users who work in a call center, based on job title.

It is not necessary to have both a DID and a non-DID range. If you only need one range, use the DID range. You can, however, still use two ranges if desired.

- ♦ How will you determine the correct phone type for a user?

The phone type must be automatically assigned by the policies, based on business rules about user information such as job title. For example, you could specify that users with the job title of Administrative Assistant receive the phone type 83424.

- ♦ Do you want to set phone use restrictions on extensions for certain users?

You can set restrictions on time of day usage, long distance calls, and international calls based on business rules and attributes of the user. For example, you can use criteria such as job title or the user object's placement in the tree.

- ♦ What user events do you want to automate with the driver, and what do you want the automated response to be?

The sample configuration demonstrates automated responses for certain user events, such as updating the display name in the PBX when the name of a user changes.

- ◆ Do you want the driver to manage all existing users, or just provision new users? If you want the driver to manage existing users, some manual setup is necessary.
The Avaya PBX driver can manage new users. It can also manage existing users.
See [Section 10.3, “Managing Existing Users,” on page 80](#).
- ◆ What do you want to use for the display name for an extension?
Determine what information from a user object you want the policies to use when creating the display name.
- ◆ How do you want to handle extensions that are not assigned to a particular user, such as conference rooms or lab extensions?
The driver does not require entities that have phone numbers to be represented in eDirectory. The driver can perform work orders in the PBX regardless of whether there is an object in eDirectory that should receive updates to phone information. But if you want to use the driver to update phone information in eDirectory for entities other than a user, one option is to create eDirectory objects for those rooms or other entities, so that they can be identified by an ObjectID. Then you can customize the driver to update them the same way it can update user objects.
- ◆ What should you consider when implementing the driver with a work order database?
Keep in mind that you could use more than one container for work orders, such as one for work orders created in eDirectory manually, and one for work orders that come from the work order database.

2.3 Planning Driver and Replica Placement on Your Servers

For each server where you install Identity Manager and run the driver, you need to have a master or read/write replica of all the users you want to manage.

Installation

To complete the installation of the Identity Manager Driver for Avaya PBX, perform the following steps:

- ◆ Install Identity Manager and the driver for Avaya PBX, as explained on [Section 3.2, “Perparing to Install,” on page 29](#)
- ◆ Create PBX Site objects, one for each PBX
- ◆ Create an instance of the driver using one of the sample configurations

This section outlines how to perform these steps through the following sections:

- ◆ [Section 3.1, “Prerequisites,” on page 29](#)
- ◆ [Section 3.2, “Perparing to Install,” on page 29](#)
- ◆ [Section 3.3, “Installing the Avaya Driver During Identity Manager Installation,” on page 30](#)
- ◆ [Section 3.4, “Importing the Driver Configuration in iManager,” on page 31](#)
- ◆ [Section 3.5, “Installing the Avaya Driver through Designer,” on page 34](#)
- ◆ [Section 3.6, “Activating the Driver,” on page 36](#)

3.1 Prerequisites

- ❑ Novell® Identity Manager 3.5.1 with the latest patches and product updates.
- ❑ Software required by Identity Manager 3.5.1.
For example, the correct version of iManager, eDirectory™, and NMAS™.
- ❑ Avaya PBX software versions 9, 10, 11, or 12.
- ❑ The eDirectory administrator username and password, so you can log in during the installation to allow schema extension. The schema extensions are described in [Appendix A, “Schema for PBX Management,” on page 101](#).
- ❑ An eDirectory server with a master or read/write replica of all the objects that will be affected by PBX changes.
- ❑ The items in [Section 3.2, “Perparing to Install,” on page 29](#).

3.2 Perparing to Install

- ◆ [Section 3.2.1, “Creating Containers,” on page 29](#)
- ◆ [Section 3.2.2, “Use Emulation,” on page 30](#)
- ◆ [Section 3.2.3, “Installing the Driver Preparation,” on page 30](#)

3.2.1 Creating Containers

Before you install, you need to specify or create some containers when importing the driver configuration:

- ◆ Container for holding the DirXML-pbxSite objects

- ♦ Container for holding the DirXML-nwoWorkOrder objects
- ♦ Container for holding the [DirXML-pbxExtension](#) objects, if you are using them
The sample base configuration uses these objects, but in a production environment you would usually transform events for [DirXML-pbxExtension](#) objects into events for User objects.
- ♦ Container for holding the DirXML-pbxAudixSubscriber objects, if you are using them

You should restrict rights to these containers so that only authorized administrators can change these containers or the objects they hold.

These containers and objects are also described in [Section 2.1, “Planning Issues for All Configurations,” on page 25](#) and [Appendix A, “Schema for PBX Management,” on page 101](#).

3.2.2 Use Emulation

This driver is different from other Identity Manager drivers because it provides emulation capabilities. Creating a robust test environment can be a challenge, because the only PBX available in an environment might be the production PBX. To solve this problem, we recommend using the driver’s emulation mode. See [“What Is Emulation Mode and Why Should I Use it?” on page 21](#) for more information before installing the driver.

3.2.3 Installing the Driver Preparation

You need to install the following:

- ♦ The driver shim (`Avayashim.jar`). Install it on the server where Identity Manager is installed or on a server where you will use Remote Loader to run the driver.
- ♦ The driver policies for the driver (`AvayaPBXShip-IDM3_5_0-V2.xml` and `AvayaUser-IDM3_5_0-V2.xml`) and the iManager plug-ins for using work orders. Install them on the iManager server.

If Identity Manager and iManager are on the same server, you only need to run the install program once. Otherwise, you need to run the install program on servers that are running the different components (Remote Loader, Identity Manager, iManager).

- 1 Install the Avaya driver from the Novell Identity Manager 3.5.1 CD. This is covered in [Section 3.3, “Installing the Avaya Driver During Identity Manager Installation,” on page 30](#).
- 2 Review the information about possible configurations in the following sections:
 - ♦ [Chapter 5, “Base Configuration,” on page 41](#)
 - ♦ [Chapter 7, “Workforce Tree Configuration,” on page 49](#)
 - ♦ [Chapter 8, “Work Order Database Configuration,” on page 57](#)
- 3 Continue with [Section 3.4, “Importing the Driver Configuration in iManager,” on page 31](#) or [Section 3.5, “Installing the Avaya Driver through Designer,” on page 34](#).

3.3 Installing the Avaya Driver During Identity Manager Installation

During the installation of Identity Manager, you can select multiple components to install. You install the driver as part of the Novell Identity Manager 3.5.1 installation program. For installation

instructions, refer to the “Installing Identity Manager” in the *Identity Manager 3.5.1 Installation Guide*. This procedure walks you through installing the Metadirectory engine with the Avaya driver and its driver utilities, including the Remote Loader installation procedure. If you are upgrading the driver, see [Chapter 4, “Upgrading the Driver,” on page 37](#).

NOTE: The Avaya driver is not supported to run local on a NetWare® server. Check the Avaya documentation for supported platforms.

Importing the driver configuration creates the driver object. After you have imported the configuration, you can use iManager to configure and manage the driver. See [Section 3.4, “Importing the Driver Configuration in iManager,” on page 31](#) or [Section 3.5, “Installing the Avaya Driver through Designer,” on page 34](#) for instructions on how to configure the driver.

3.4 Importing the Driver Configuration in iManager

The Create Driver Wizard helps you import the basic driver configuration file. This file creates and configures the objects and policies needed to make the driver work properly.

- 1 In Novell® iManager, click *Identity Manager Utilities > New Driver*.
- 2 Select a driver set.
If you place this driver in a new driver set, you must specify a driver set name, context, and associated server.
- 3 Select *Import a Driver Configuration from the Server (.XML file)*, then select *AvayaPBXShip-IDM3_5_0-V2.xml*.
The driver configuration files are installed on the Web server when you install Identity Manager. During the import, you are prompted for the driver’s parameters and other information.
- 4 Specify parameter values. See [Table 3-1 on page 31](#) for more information.
- 5 Click *Import*.
When the import is finished, you should define security equivalences and exclude administrative roles from replication.
The driver object must be granted sufficient eDirectory™ rights to any object it reads or writes. You can do this by granting Security Equivalence to the driver object. The driver must have Read/Write access to users, resources, and distribution lists. Normally, the driver should be given security equal to Admin.
- 6 Review the driver objects in the *Summary* page, then click *Finish*.

Table 3-1 Specify Values for These Parameters

Parameter Name	Parameter Description
Driver name	The actual name you want to use for the driver.
Emulation Mode:	Will this configuration be initially setup to use emulation mode? Emulation mode provides you with the ability to test live data but not write it out to your phone systems. Default is <i>Emulation</i> .

Parameter Name	Parameter Description
Polling Method:	Select the method the driver should use to poll for changes. <i>By Interval</i> means the driver will poll every so often for a given interval specified in minutes. <i>By Time</i> means the driver will poll only at a specified time. Or, you may choose to do both.
Driver is Local/Remote	Configure the driver for use with the Remote Loader service by selecting the Remote option, or select Local to configure the driver for local use. If Local is selected, you can skip the remaining parameters.
Site Container:	Specify the container where site objects will be placed in the directory. For example: <code>ou=PbxSite.o=MyOrganization</code> .
Work Orders Container:	Specify the container where work order objects will be placed in the directory. For example: <code>ou=WorkOrders.o=MyOrganization</code> .
Users Container:	Specify the container that will hold the User objects in the directory. For example: <code>ou=Users.o=MyOrganization</code> .
PBX Name:	Specify the name of the PBX site object. This is the same name you use for the creation of the PBX site object in the directory. For example: <code>[ProvoPBX]</code>
Driver will Manage Avaya Extensions:	Select whether this driver will be managing Avaya Extensions. Default is <i>Yes</i> .
Driver will Manage Audix Subscribers:	Select whether this driver will be managing Audix* subscribers. Default is <i>Yes</i> .
Extensions Container:	Specify the container where extension objects will be placed in the directory. For example: <code>ou=Extensions.o=MyOrganization</code> .
Subscriber Container:	Specify the container where subscriber objects will be placed in the directory. For example: <code>ou=Subscribers.o=MyOrganization</code> .
LDAP Host IP Address:	Specify the LDAP host IP address to be used in emulation mode, for example: <code>[151.65.137.1]</code> .
LDAP Host Port Number:	Specify the LDAP host port number to be used in emulation mode, for example: <code>[389]</code> .
LDAP Host User DN:	Specify the LDAP host users DN used for authentication to the emulation LDAP system, for example: <code>[cn=Admin,o=Org]</code> .
LDAP Host Password:	Specify the LDAP host users password to be used in emulation mode, for example: <code>[Novell]</code> . Confirm the password.
Extensions Container:	Specify the LDAP host container that will be used for the emulation of extensions, for example: <code>[ou=PbxEmulationExtensions,o=Org]</code> .
Poll Interval (minutes):	Specify a poll interval, in minutes, for the driver to use. For example, 120.

The additional driver parameters are set to default values during the import process, but they can be modified in iManager (by clicking the Driver Configuration tab on the driver object).

Table 3-2 *Additional Configuration Information*

Import Prompt	Description
Poll Time	<p>If you choose to use a polling time, this prompt is displayed when you click <i>Next</i>.</p> <p>Select a poll time for the driver to use. For example, 12:00 AM.</p>
Remote Host Name and Port	<p>For remote driver configuration only. If you specify Remote in the Driver is Local/Remote prompt, this prompt is displayed when you click <i>Next</i>.</p> <p>Specify the host name or IP address and port number where the Remote Loader Service has been installed and is running for this driver. The default port is 8090.</p>
Driver Password	<p>For remote driver configuration only. If you specify Remote in the Driver is Local/Remote prompt, this prompt is displayed when you click <i>Next</i>.</p> <p>The Driver object password is used by the Remote Loader to authenticate itself to the Identity Manager server. It must be the same password that is specified as the Driver object password on the IDM Remote Loader.</p>
Remote Password	<p>For remote driver configuration only. If you specify Remote in the Driver is Local/Remote prompt, this prompt is displayed when you click <i>Next</i>.</p> <p>The Remote Loader password is used to control access to the Remote Loader instance. It must be the same password that is specified as the Remote Loader password on the IDM Remote Loader.</p>

If you use the Emulation mode, specify the following parameters when importing the driver.

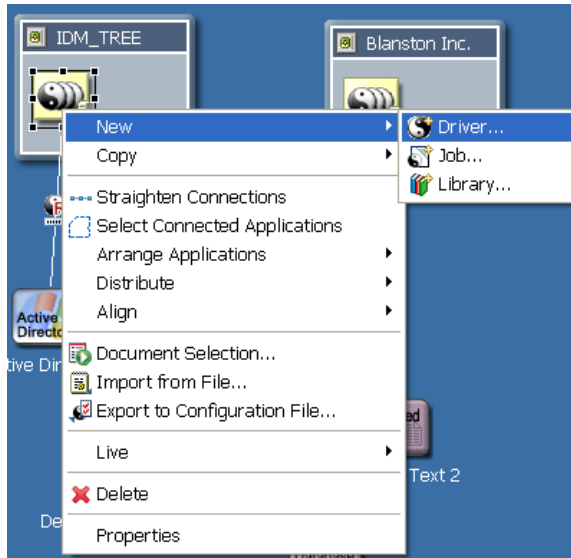
Table 3-3 *Specify Parameters for Emulation Mode*

Import Prompt	Description
IP Address for LDAP Host for Emulation	<p>For emulation, the driver needs to access an LDAP host (eDirectory or any other host).</p> <p>Specify the IP address of the LDAP server.</p>
Port of LDAP Host	<p>Specify the port number of the LDAP server used for emulation.</p>
DN of the Login User for Emulation	<p>For emulation, the driver needs to know which username has access to the LDAP server. This user must have rights to read/write to the extension containers on the server.</p> <p>Specify the name of the user for the LDAP server.</p>
User Password for Emulation	<p>Specify the password of the user who accesses the LDAP server.</p>
Extension Container DN for Emulation	<p>Specify the DN of the container where the driver will add or modify extension objects (this container must already exist on the LDAP server.)</p>

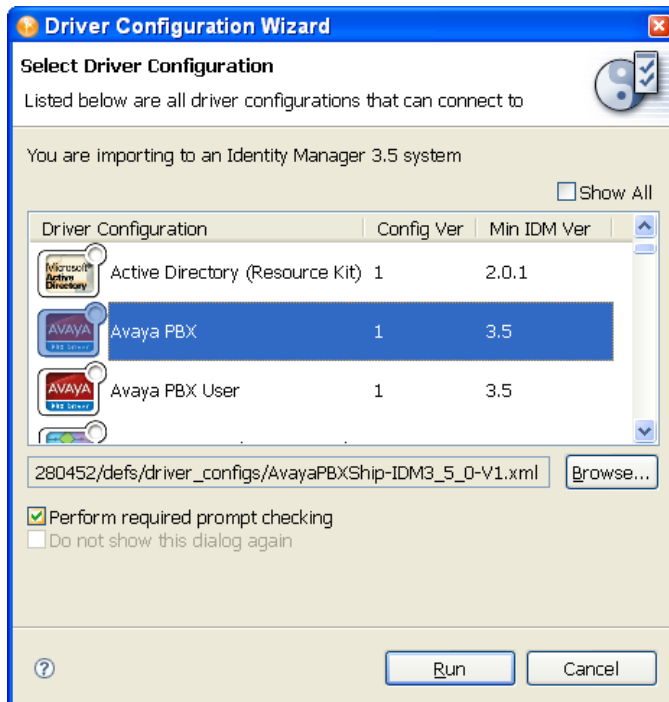
3.5 Installing the Avaya Driver through Designer

Designer has a Driver Configuration Wizard to help you import and configure the Avaya driver configuration file. This file creates and configures the objects and policies needed to make the driver work properly.

- 1 In Designer, right-click the driver set where you plan to install and configure the Avaya driver, then select *New > Driver*.



- 2 In the Connecting to Application window, select the `AvayaPBXShip-IDM3_5_0-V2.xml` file, select *Perform required prompt checking*, then click *Run*.



3 Specify values for the following parameters:

Table 3-4 *Specify Values for These Parameters*

Parameter Name	Parameter Description
Driver name	The actual name you want to use for the driver.
Emulation Mode:	Select whether this configuration should be initially set up to use emulation mode. Emulation mode provides you with the ability to test live data but not write it out to your phone systems. The default is <i>Emulation</i> .
Polling Method:	Select the method the driver should use to poll for changes. <i>By Interval</i> means the driver polls at a given interval specified in minutes. <i>By Time</i> means the driver polls only at a specified time. You can choose to do both.
Driver is Local/Remote	Configure the driver for use with the Remote Loader service by selecting the <i>Remote</i> option, or select <i>Local</i> to configure the driver for local use. If <i>Local</i> is selected, you can skip the remaining parameters.
Site Container:	Specify the container where site objects will be placed in the directory. For example: <code>ou=PbxSite.o=MyOrganization</code> .
Work Orders Container:	Specify the container where work order objects will be placed in the directory. For example: <code>ou=WorkOrders.o=MyOrganization</code> .
Users Container:	Specify the container that will hold the User objects in the directory. For example: <code>ou=Users.o=MyOrganization</code> .
PBX Name:	Specify the name of the PBX site object. This will be the same name you use for the creation of the PBX site object in the directory. For example: <code>[ProvoPBX]</code>
Driver will Manage Avaya Extensions:	Select whether this driver will be managing Avaya Extensions. The default is <i>Yes</i> .
Driver will Manage Audix Subscribers:	Select whether this driver will be managing Audix subscribers. The default is <i>Yes</i> .
Extensions Container:	Specify the container where extension objects will be placed in the directory. For example: <code>ou=Extensions.o=MyOrganization</code> .
Subscriber Container:	Specify the container where subscriber objects will be placed in the directory. For example: <code>ou=Subscribers.o=MyOrganization</code> .
LDAP Host IP Address:	Specify the LDAP host IP address to be used in emulation mode, for example: <code>[151.65.137.1]</code> .
LDAP Host Port Number:	Specify the LDAP host port number to be used in emulation mode, for example: <code>[389]</code> .
LDAP Host User DN:	Specify the LDAP host users DN used for authentication to the emulation LDAP system, for example: <code>[cn=Admin,o=Org]</code> .
LDAP Host Password:	Specify the LDAP host users password to be used in emulation mode, for example: <code>[Novell]</code> . Confirm the password.
Extensions Container:	Specify the LDAP host container that will be used for the emulation of extensions, for example: <code>[ou=PbxEmulationExtensions,o=Org]</code> .

Parameter Name	Parameter Description
Poll Interval (minutes):	Specify a poll interval, in minutes, for the driver to use. For example, 120.

The additional driver parameters are set to default values during the import process, but they can be modified in iManager (by clicking the Driver Configuration tab on the driver object).

3.6 Activating the Driver

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

For more information, refer to “[Activating Novell Identity Manager Products](#)” in the *Identity Manager 3.5.1 Installation Guide*.

Upgrading the Driver

If you have been using a previous version of the driver, follow these instructions instead of the ones in [Chapter 3, “Installation,”](#) on page 29.

- ♦ [Section 4.1, “Changes in Policy Architecture,”](#) on page 37
- ♦ [Section 4.2, “Upgrading the Driver in Designer,”](#) on page 37
- ♦ [Section 4.3, “Upgrading the Driver in iManager,”](#) on page 40

4.1 Changes in Policy Architecture

Identity Manager 3.5 and 3.5.1 contain a new policy architecture, which affects how drivers reference policies. While the 3.5.1 driver architecture offers increased functionality in the Identity Manager 3.5.1 environment, the 3.0.x Metadirectory engine cannot run 3.5.1 driver configurations.

However, Identity Manager 3.5 and 3.5.1 can run 3.0x driver configurations. If you have 3.0.x driver configurations associated with both 3.0.x and 3.5.1 Metadirectory engines, do not upgrade the 3.0.x drivers. The 3.0.x driver configurations work in a 3.5.1 environment, but they don't have the increased functionality that Identity Manager 3.5 and above affords. When 3.0.x driver configurations are only associated with a 3.5 or later Metadirectory engine, you should then upgrade the 3.0.x drivers to 3.5.1.

For more information on policy architecture and upgrading drivers to 3.5.1, see [“Upgrading Identity Manager Policies”](#) in *Understanding Policies for Identity Manager 3.5.1*.

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

4.2 Upgrading the Driver in Designer

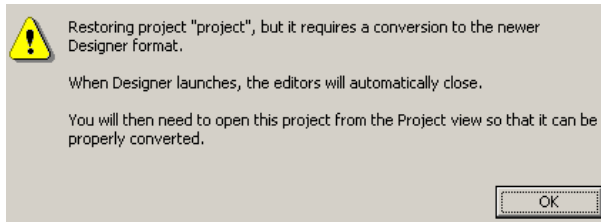
- 1 Make sure you have updated your driver with all the patches for the version you are currently running.

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

- 2 Back up the driver. See [Chapter 12, “Backing Up the Driver,”](#) on page 89 for instruction on how to back up the driver.
- 3 Install Designer version 2.0 or above, then launch Designer.

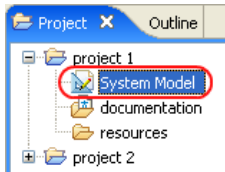
If you had a project open in Designer when you upgraded Designer, proceed to [Step 4](#). If you didn't have a project open in Designer when you upgraded Designer, skip to [Step 5](#).

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

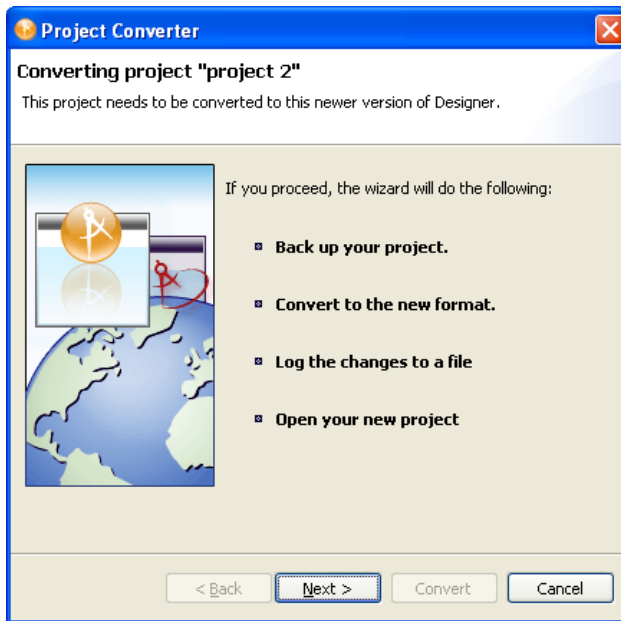


Designer closes the project to preform the upgrade.

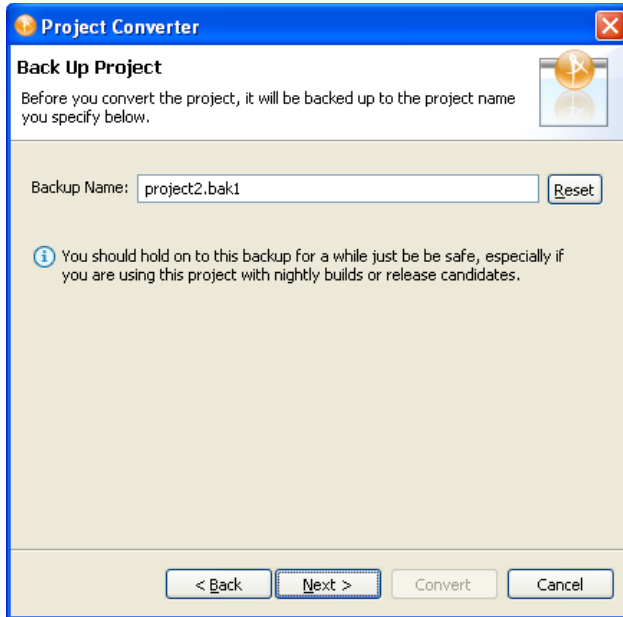
- 5 In the Project view, double-click *System Model* to open and convert the project.



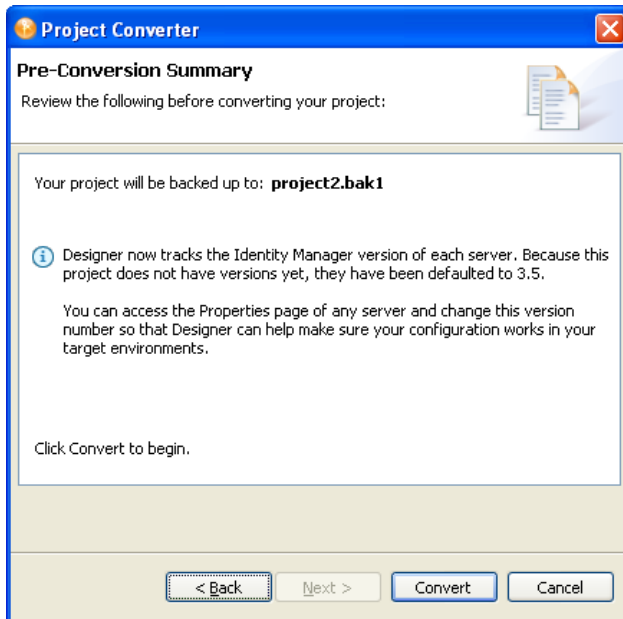
- 6 Read the Project Converter message explaining that the project is backed up, converted to the new format, changes logged to a file, and the new project is opened, then click *Next*.



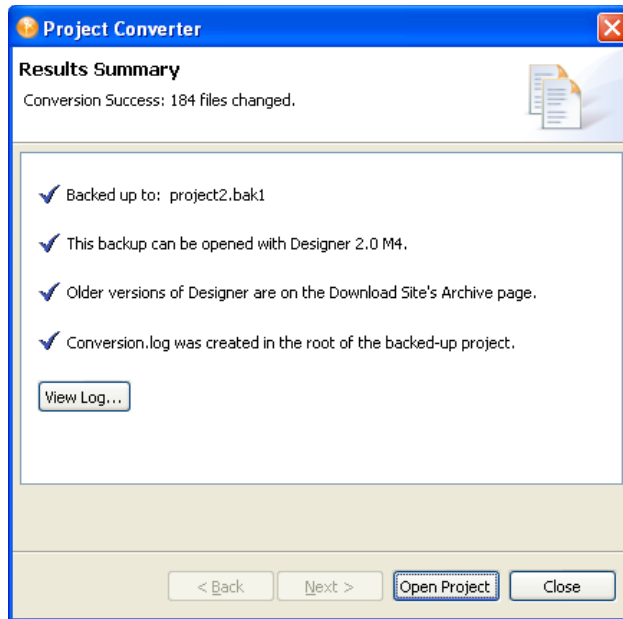
7 Specify the name of the backup project name, then click *Next*.



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



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



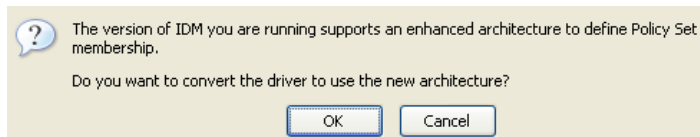
If you want to view the log file that is generated, click *View Log*.

4.3 Upgrading the Driver in iManager

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

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

- 2 Back up the driver. See [Chapter 12, “Backing Up the Driver,” on page 89](#) for instruction on how to back up the driver.
- 3 Verify that Identity Manager 3.5.1 has been installed and you have the current plug-ins installed, then launch iManager.
- 4 Click *Identity Manager > Identity Manager Overview*.
- 5 Click *Search* to find the Driver Set object, then click the driver you want to upgrade.
- 6 Read the message that is displayed, then click *OK*.



- 7 If there is more than one driver to upgrade, repeat [Step 2](#) through [Step 6](#).

Base Configuration

- ♦ [Section 5.1, “How the Base Configuration Works,” on page 41](#)
- ♦ [Section 5.2, “Planning for the Base Configuration,” on page 46](#)
- ♦ [Section 5.3, “Setting Up the Base Configuration,” on page 46](#)

The base configuration (`AvayaPBXShip-IDM3_5_0-V2.xml`) is a sample configuration to demonstrate the most basic functionality of the driver. It shows how the driver can manage PBX extensions using work order objects in eDirectory™.

This configuration would be appropriate to import and configure in a test environment for instructional purposes, to help you learn how to configure the solution you need. It is not meant to be an out-of-the-box solution or as a starting point for installation setups.

The rules and policies in the base configuration are set up so that you can create work order objects (using a new object class, `nwoWorkOrder`) in a Work Order container, and the driver will perform the work orders in the PBX and show the results by creating a `pbxExtension` object and updating the `nwoWorkOrder` object in eDirectory.

Most real-life implementations would want User objects to be updated when an extension is assigned or changed, but to keep the configuration sample as simple as possible, the base configuration does not include rules to do this. Instead, `pbxExtension` objects are used to represent extensions, and User objects are not affected by the changes.

To see how user objects can be involved in the process, review the workforce tree configuration, explained in [Chapter 7, “Workforce Tree Configuration,” on page 49](#).

Similarly, the base configuration does not demonstrate a connection between eDirectory and a work order database. In implementations that have an existing work order database, you should connect to it so that the Identity Manager Driver for Avaya PBX can send and receive work order data. This connection could be made using another driver such as the IDM Driver for JDBC. This kind of implementation is described in [Chapter 8, “Work Order Database Configuration,” on page 57](#).

5.1 How the Base Configuration Works

[Figure 5-1](#) shows what happens in the base configuration if you create an `nwoWorkOrder` object for installing a new extension.

1. You create an `nwoWorkOrder` object in eDirectory by using the Install task. In this example, the flags are set on the work order for `SendToPublisher` and `DoItNow`.
2. The Subscriber channel is notified by the Metadirectory engine that a new `nwoWorkOrder` object has been created in eDirectory.
3. The Subscriber creates the Identity Manager association for the object, and sends the work order to the Publisher.

(If the `SendToPublisher` flag is not set, the Subscriber does not “wake up” Publisher channel to send the work order to the Publisher. Instead, the Publisher reads the work order the next time it polled for work orders.)

4. The Publisher performs the work order immediately, because the `DoItNow` flag is set.

(If the DoItNow flag is not set, the Publisher doesn't perform the action until the date specified in the DueDate attribute.)

5. After successfully configuring the extension in the PBX, the Publisher writes "configured" in the Status attribute of the work order.
6. The Publisher creates a pbxExtension object in eDirectory to represent the new extension that has been configured.

After the driver completes the work order in the PBX, the PBX admin can complete any manual tasks required, such as plugging in the phone and punching in the activation code for hot jacks, or cross-connecting the wiring, etc., for cold jacks.

The following figures describe the base configuration. **Figure 5-1** shows an illustration of how the base configuration generally works. **Figure 5-2** is a flowchart of how the configuration works for an Install work order with the DoItNow flag set to true, and **Figure 5-3** is a flowchart showing how the configuration works for an Install work order with the DoItNow flag not set.

Figure 5-1 Graphical Representation of Base Configuration

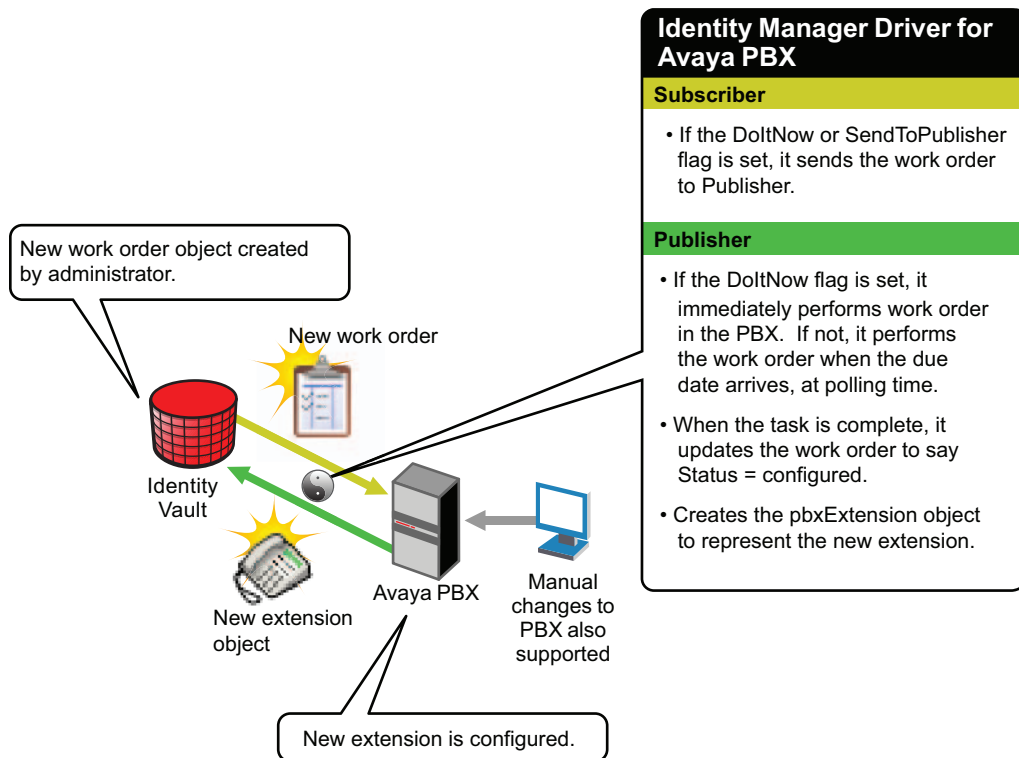


Figure 5-2 Flowchart of Base Configuration for Work Order with DoItNow Flag Set to True

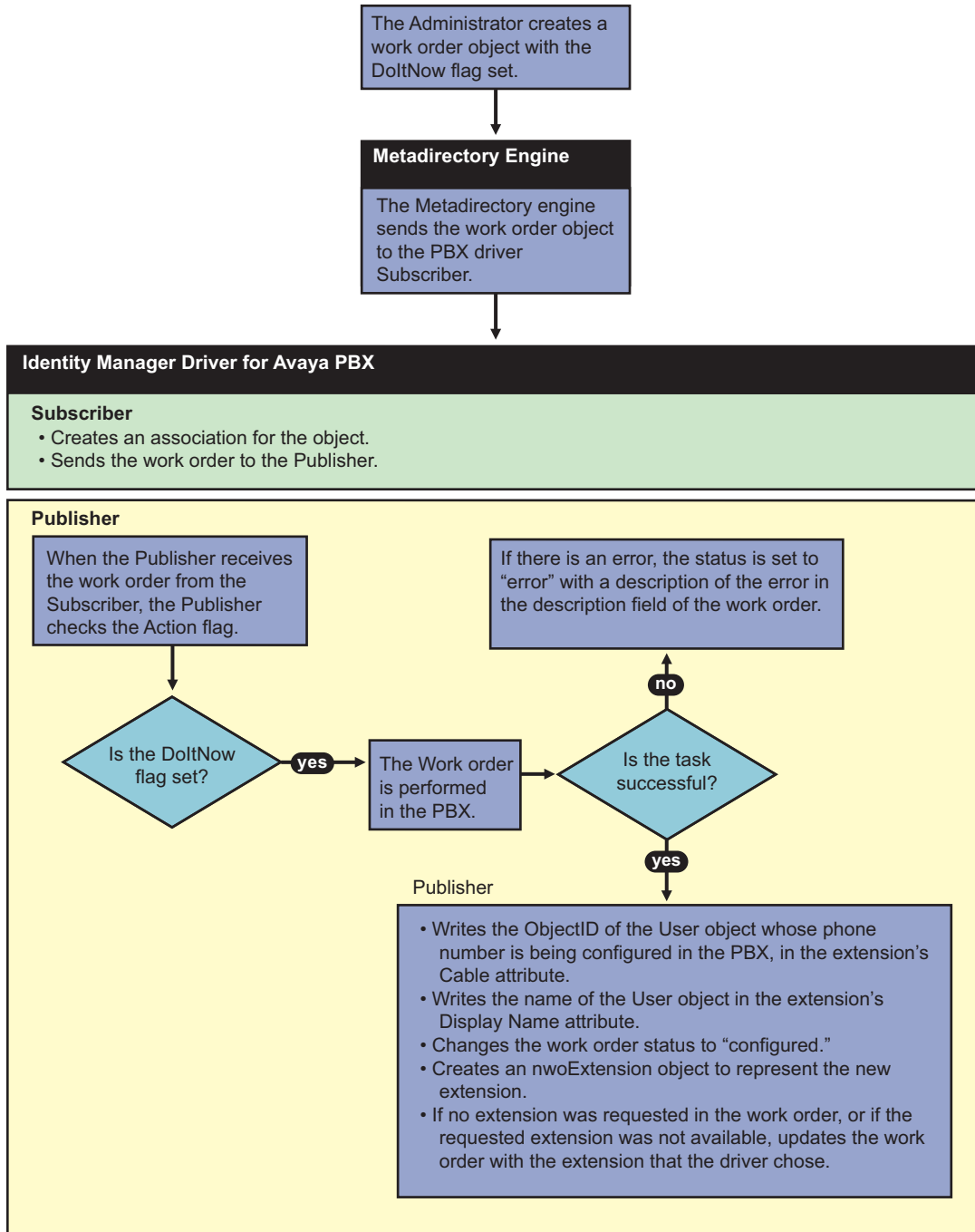
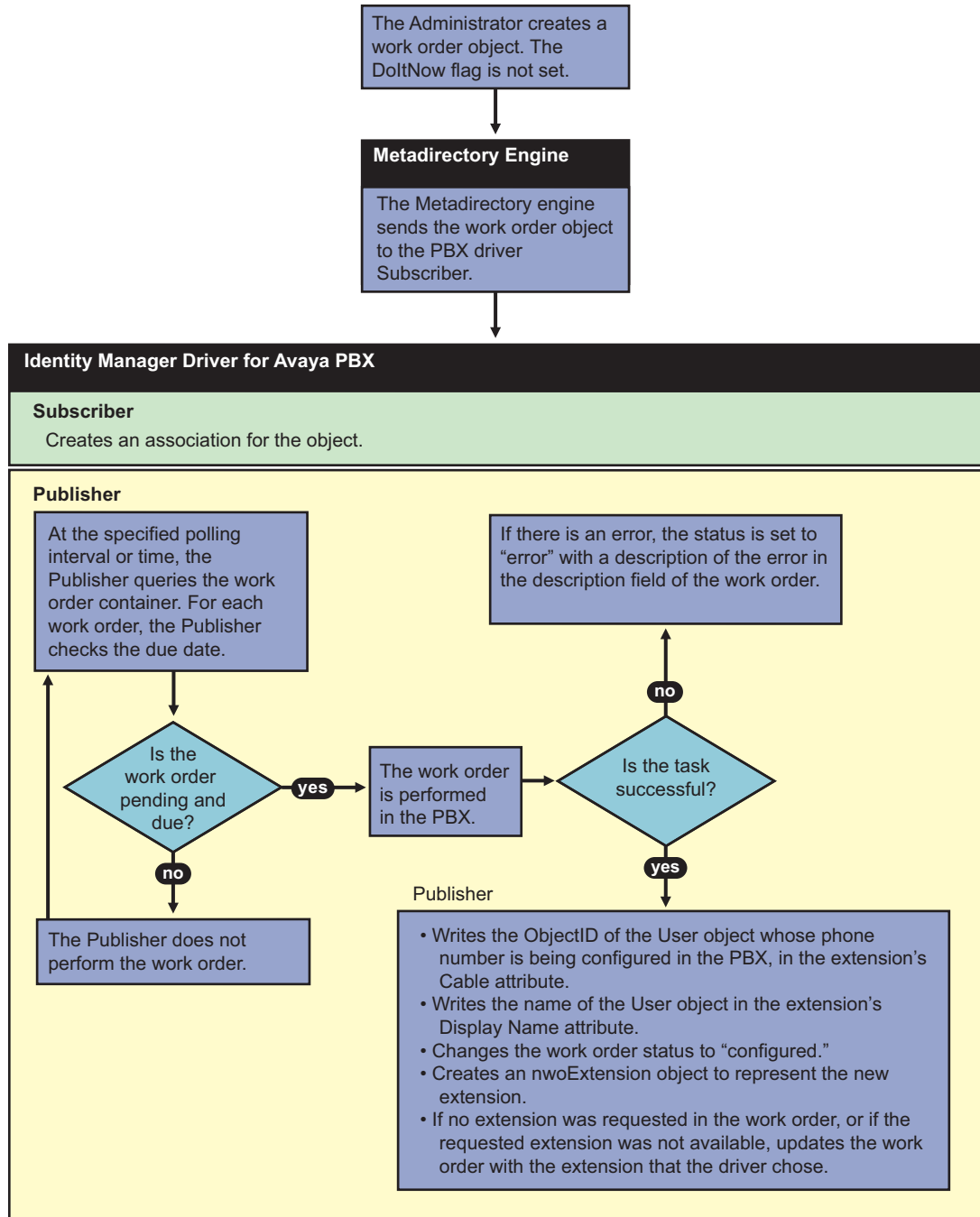


Figure 5-3 Flowchart of Base Configuration for Work Order with DoItNow Flag Not Set



5.1.1 How the Subscriber Channel Is Configured

In the base configuration, the Subscriber channel processes only events that pertain to work orders.

You must create pbxSite objects for each site in Identity Vault, so the driver knows how to contact each PBX, but the driver queries for this information only at startup. The Subscriber does not listen for events pertaining to pbxSite objects, so if changes are made to these objects you must stop and restart the driver for the changes to be recognized by the driver.

For many drivers, the Subscriber performs changes in the third-party application in response to events in Identity Vault. However for this driver, the Publisher is the agent that performs work orders in the PBX.

Table 5-1 *Configuring the Subscriber Channel*

Rule or Policy	What it does
Subscriber Filter	Allows only events for nwoWorkOrder objects to be processed.
Event Transformation	Not used in the sample configuration.
Matching Rule	Not used in the sample configuration.
Create Rule	<p>Contains rules only for work order objects.</p> <p>Requires values for the following attributes on a work order object.</p> <ul style="list-style-type: none"> ◆ PBXName ◆ nwoStatus ◆ nwoSendtoPublisher ◆ nwoDoltNow ◆ nwoAction ◆ nwoDisplayName <p>If the values are not present, the work order is not sent to the Publisher, and therefore is not performed in the PBX.</p> <p>For a description of these attributes, see Section A.2, “DirXML-nwoWorkOrder Object,” on page 103.</p>
Placement Rule	Maps work orders from the work order container you specified to the PBX driver. This mapping is necessary so that the Subscriber can check the work orders to see whether the DoltNow flag is set to True.
Command Transformation	Not used in the sample configuration.
Schema Mapper	<p>Maps the eDirectory namespace to the PBX namespace.</p> <p>Handles mapping of work orders, extensions, and PBX site objects.</p>
Output Transformation	Not used in the sample configuration.

5.1.2 How the Publisher Channel Is Configured

Through the Publisher channel, the Identity Manager Driver for Avaya PBX queries the PBX for information about extensions. The Publisher performs tasks in the Avaya PBX, rather than the Subscriber. For a general overview of what the Publisher does, see [Section 1.5.2, “Overview of Driver Functionality,”](#) on page 17.

Table 5-2 *Configuring the Publisher Channel*

Rule or Policy	What it does
Input Transformation	Not used in the sample configuration.

Rule or Policy	What it does
Schema Mapper	Maps the PBX namespace to the eDirectory namespace. Handles mapping of work orders, extensions, and PBX site objects.
Event Transformation	Not used in the sample configuration.
Publisher Filter	Allows only events for nwoWorkOrder and pbxExtension objects to be processed.
Matching Rule	Not used in the sample configuration
Placement Rule	Places workOrder objects in the correct container as defined in the driver's configuration parameters. Places nwoExtension objects in the correct container.
Command Transformation	Not used in the sample configuration.

5.2 Planning for the Base Configuration

See [Chapter 2, “Planning,” on page 25](#).

5.3 Setting Up the Base Configuration

Follow the instructions in [Chapter 3, “Installation,” on page 29](#). When you are creating a driver object, use the sample configuration file named `AvayaPBXShip-IDM3_5_0-V2.xml`.

Activating the Driver

6

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

To activate the driver, see “[Activating Novell Identity Manager Products](#)” in the *Identity Manager 3.5.1 Installation Guide*.

Workforce Tree Configuration

The workforce tree configuration demonstrates how the driver can be configured to provision users in the PBX system by doing the following:

- ◆ Assigning an extension to new users by creating a work order when a new user is added
- ◆ Enforcing business policies about phone use restrictions, based on user attributes
- ◆ Performing work orders to install, modify, move, disable, or disconnect existing extensions, based on user events such as change in location or job status.
- ◆ Updating user objects with new phone extension information to reflect changes made in the PBX.

Like the base configuration, the workforce tree configuration is meant to be instructional, and to demonstrate what the driver can do. It is not meant to be an out-of-the-box solution.

In contrast with the base configuration, rules are in place in the workforce tree configuration to maintain the relationships between users in the workforce tree and extensions in the PBX. For example, changes to users cause appropriate work orders to be created. And, when an extension is assigned in the PBX, the phone number is added to the user object's attributes.

In some implementations, you should connect to an existing work order database so that the Identity Manager Driver for Avaya PBX can perform work orders from a system that is already in place. The workforce tree configuration does not demonstrate this functionality. This aspect of how the Avaya PBX driver is explained in [Chapter 8, “Work Order Database Configuration,” on page 57](#).

In this section:

- ◆ [Section 7.1, “How the Workforce Tree Configuration Works,” on page 49](#)
- ◆ [Section 7.2, “Planning for the Workforce Tree Configuration,” on page 56](#)
- ◆ [Section 7.3, “Setting Up the Workforce Tree Configuration,” on page 56](#)
- ◆ [Section 7.4, “Using Log/Reporting Functionality to Report Warnings,” on page 56](#)

7.1 How the Workforce Tree Configuration Works

The Subscriber adds to the work order the ID of the user object for which the phone number is being provisioned. This can be an ID of your choice, for example, Employee ID or Social Security number. The sample configuration uses Workforce ID. When the Publisher has configured the extension, it updates the user object with the extension number. The driver puts the Object ID in the Cable field of the extension in the PBX. This field has a character limit of 5 characters.

Figure 7-1 Graphical Representation of Workforce Tree Configuration

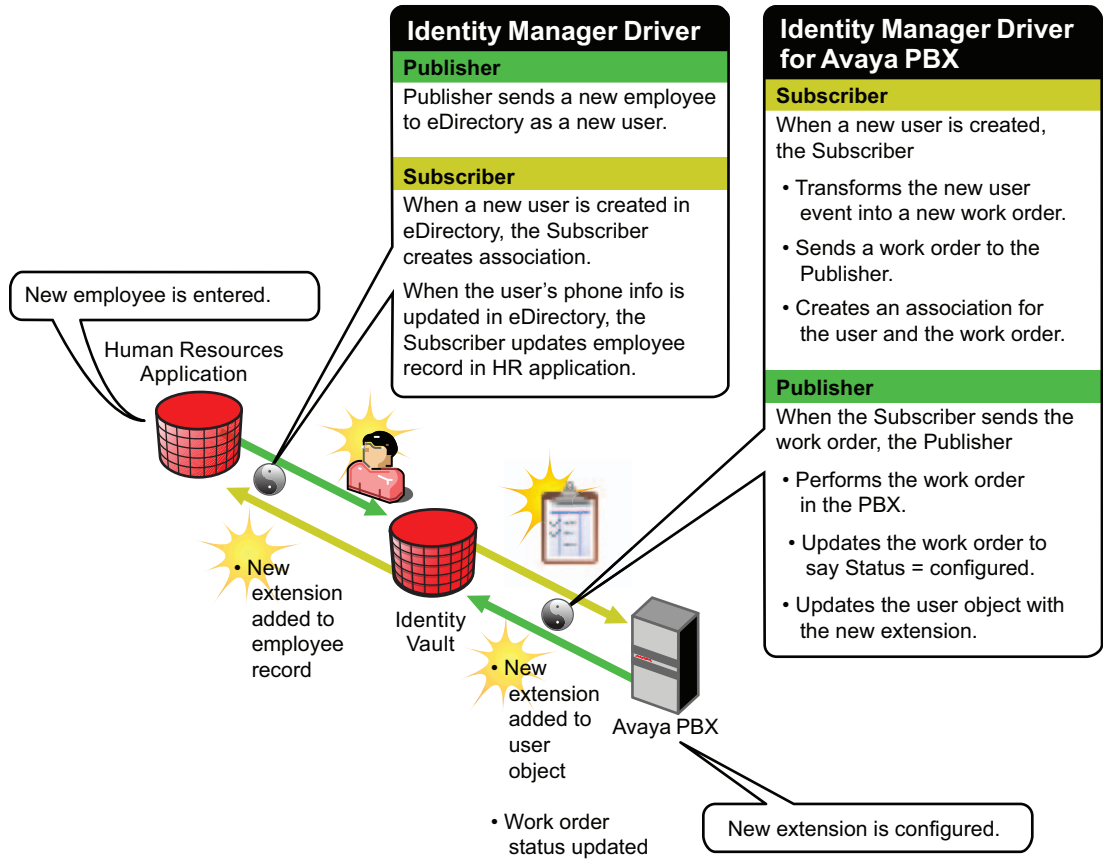
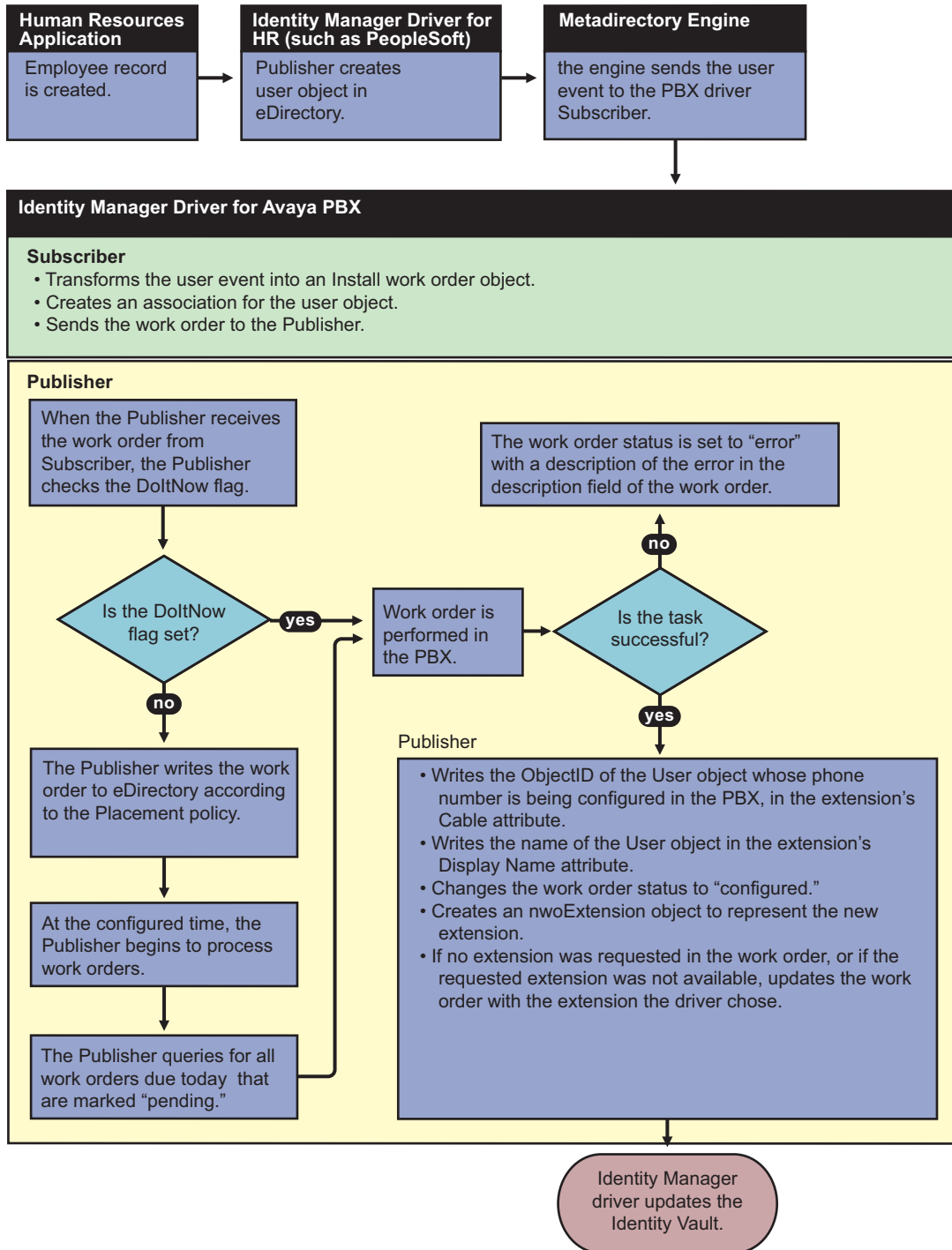


Figure 7-2 Flowchart of Workforce Tree Configuration

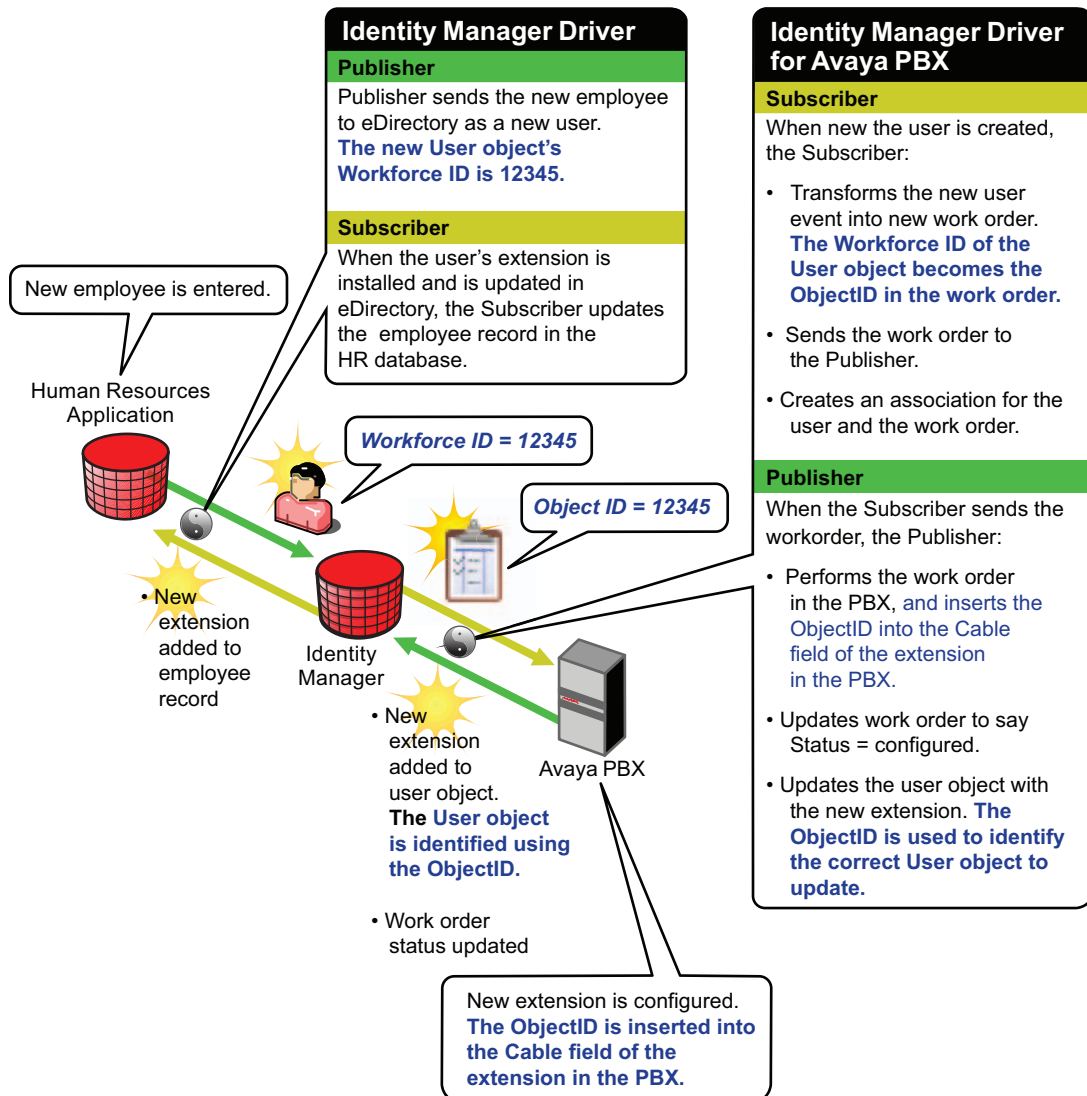


7.1.1 How the Driver Uses the ObjectID to Identify and Update Users

The following figure shows where the ObjectID comes from and how it is used by the driver to update users. In this diagram you can trace the role of the ObjectID; the user’s ID becomes the

ObjectID of the work order, and is entered into the PBX as an attribute of the extension. After the task is complete, the ObjectID is used to identify the user object that needs to be updated.

Figure 7-3 How the Object ID Is Used in the Process



7.1.2 How the Subscriber Channel Is Configured

In the workforce tree configuration, the Subscriber channel listens for events that pertain to work orders and users. (PBX site objects must also exist in eDirectory™, but they are used only for the driver to query for information about how to contact each PBX. Changes to PBX objects are not processed by the Subscriber; instead you must restart the driver for changes to be recognized.)

In the workforce tree configuration, a work order can be created or a user can be created, which then causes Identity Manager to create a work order.

Table 7-1 *Configuring the Subscriber Channel*

Rule or Policy	What it does
Subscriber Filter	Allows only events for nwoWorkOrder and User objects to be processed.
Event Transformation	Not used in the sample configuration.
Matching Rule	Not used in the sample configuration.
Create Rule	<p>Contains rules only for nwoWorkOrder and User objects.</p> <p>For an nwoWorkOrder object, requires values for the following attributes.</p> <ul style="list-style-type: none">◆ PBXName◆ nwoStatus◆ nwoSendtoPublisher◆ nwoDoltNow◆ nwoAction◆ nwoDisplayName <p>For a User object, requires values for the following attributes.</p> <ul style="list-style-type: none">◆ Surname◆ L <p>As an example in the sample configuration, L (location) is used to provision the extension differently depending on where the user is physically located. (You could customize the policy to use L to determine which PBX site to use, or what setcor restrictions to use.)</p> ◆ Workforce ID <p>The Workforce ID is important in the sample configuration because it is used to populate the ObjectID so the user can be identified with a work order.</p> <p>If values are not present for the required attributes, the work order is not sent to the Publisher, and therefore is not performed in the PBX.</p> <p>For a description of these attributes, see Section A.2, “DirXML-nwoWorkOrder Object,” on page 103 and Section A.5, “User Objects and Their Identity Manager Associations,” on page 111.</p>
Placement Rule	<p>Maps work orders from the work order container you specified to the Identity Manager Driver for Avaya PBX. This mapping is necessary so that the Subscriber can check the work orders to see whether the DoltNow flag is set to True.</p> <p>Maps user object containers to the Identity Manager Driver for Avaya PBX. This makes sure that users, when they are created, are sent to the driver so that work orders can be created to provision their phone extensions.</p>

Rule or Policy	What it does
Command Transformation	<p>Takes user events and changes them to PBX actions.</p> <ul style="list-style-type: none"> ◆ Converts a user Add event into a work order object. ◆ Determines the PBX and node to use based on the Location attribute of the user object. (The configuration includes sample code for how to do this. You could use another user attribute instead, such as job title.) ◆ Determines the phone type based on the Location attribute of the user object. Then, determines the dupe extension to use based on the phone type. The sample configuration shows an example of mapping phone type to dupe extension.
Schema Mapper	<p>Maps the eDirectory namespace to the PBX namespace.</p> <p>Handles mapping of work orders, extensions, and PBX site objects.</p>
Output Transformation	<p>For work orders you create in the work order container, maps the phone type to the duplicate extension if the dupe extension is not indicated in the work order. (Use the PBX “template” for how to configure the phone type).</p>

7.1.3 How the Publisher Channel Is Configured

Through the Publisher channel, the Identity Manager Driver for Avaya PBX queries the PBX for information about extensions, places work order objects in the correct container, performs work orders, and sends updates to user objects.

Table 7-2 *Configuring the Publisher Channel*

Rule or Policy	What it does
Input Transformation	Not used in the sample configuration.
Schema Mapper	<p>Maps the PBX namespace to the eDirectory namespace.</p> <p>Handles mapping of work orders, extensions, and PBX site objects.</p>
Event Transformation	<p>Transforms pbxExtension Add, Modify, or Delete events into events that modify user objects.</p> <ul style="list-style-type: none"> ◆ Transforms nwoExtension Add events into user Modify events. Uses the ObjectID from the work order to identify the user so it can add the new extension to the User object's telephone number. ◆ Transforms Modify events into events to modify the preferred name or object ID. ◆ Transforms Delete events for extensions into Modify events to remove the extension from all users who have it in their phone list. This is done by a query for users, rather than by the Object ID.
Publisher Filter	Allows only events for nwoWorkOrder, User, and extension objects to be processed.
Matching Rule	Not used in the sample configuration.
Create Rule	Not used in the sample configuration.

Rule or Policy	What it does
Placement Rule	<ul style="list-style-type: none"> ◆ Places work order objects in the correct container. ◆ Places extension objects in the correct container.
Command Transformation	Not used in the sample configuration.

7.1.4 User Events That Can Trigger a Work Order

Table 7-3 *Configuring User Events*

User event from the Human Resources application	Work Order that is created by the policies	Ideas for what you could do when customizing the policies
New employee is entered in HR application	<ul style="list-style-type: none"> ◆ Install work order is created to install new extension ◆ Work order is marked DoItNow, so it is performed immediately ◆ The user is updated with the new phone extension in the Identity Vault 	You could configure the HR driver to update the user's information in the HR application.
Employee moves to a different location	No action is taken in the sample configuration.	<p>You could customize the policies to do the following:</p> <ul style="list-style-type: none"> ◆ Create a Move work order ◆ If the move requires a new extension update, give the employee a new extension ◆ Remove the old extension from the user object.
Employee's name changes	<ul style="list-style-type: none"> ◆ A Modify work order is created and is performed immediately ◆ The display name in the PBX is modified, so the users' phone shows the correct name on outgoing calls. 	
Employee goes on extended leave	No action is taken in the sample configuration.	You could customize the policies to create a Disable work order.
Employee leaves the company	No action is taken in the sample configuration.	You could customize the policies to disconnect the extension.

7.2 Planning for the Workforce Tree Configuration

See [Chapter 2, “Planning,”](#) on page 25.

7.3 Setting Up the Workforce Tree Configuration

Follow the instructions in [Chapter 3, “Installation,”](#) on page 29, and when you are creating a driver object, use the sample configuration file named `AvayaUser-IDM3_5_0-V2.xml`.

7.4 Using Log/Reporting Functionality to Report Warnings

You might want to record the warnings that occur if a user object that is created does not match the Create policy criteria. This would mean that someone is not following the business processes.

You can keep track of this by using the log file or Novell[®] Audit to capture the warning messages that are generated.

Work Order Database Configuration

As described in [Chapter 7, “Workforce Tree Configuration,” on page 49](#), you can use the Identity Vault as your work order database, using work order objects. However, if you have another work order database that you are already using to enter PBX work orders, the Identity Manager Driver for Avaya PBX can fit into that environment as well. The driver can perform work orders that are created in another application and synchronized to the Identity Vault.

For example, if personnel in your organization are accustomed to using a JDBC* database product to enter work orders for phone extensions, you could preserve the existing process for entering work orders while using the Avaya PBX driver to automate how the work orders are performed. This kind of solution would use two Identity Manager drivers, one for Avaya PBX, and one for the JDBC database.

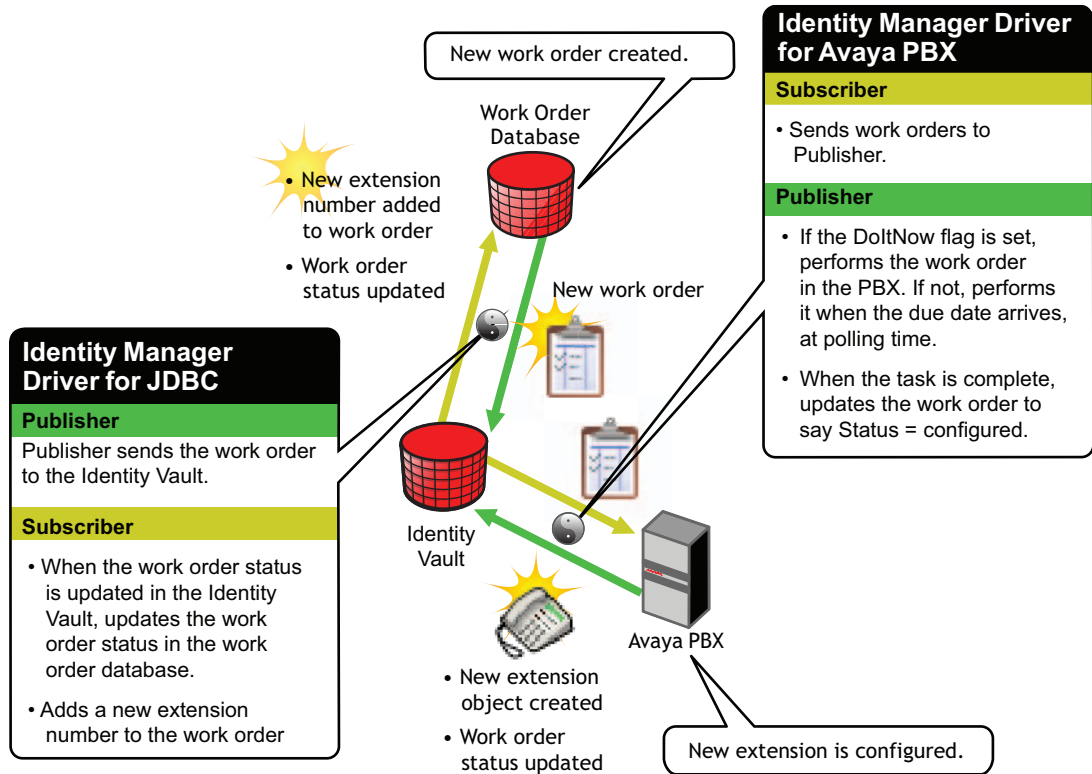
This solution could be combined with the Workforce Tree configuration example, so that work orders can have two sources:

- ♦ Automated requests triggered by user events in eDirectory, such as new extensions assigned for new employees.
- ♦ Manual requests for existing employees entered in a work order database by administrative assistants or IT personnel, such as modifying a display name or moving offices.
- ♦ [Section 8.1, “Configuring a Work Order Database Configuration,” on page 57](#)
- ♦ [Section 8.2, “About Work Order Systems,” on page 61](#)
- ♦ [Section 8.3, “Planning for the Work Order Database Configuration,” on page 61](#)
- ♦ [Section 8.4, “Setting Up the Work Order Database Configuration,” on page 61](#)
- ♦ [Section 8.5, “Using the Log/Reporting Functionality to Report Warnings,” on page 61](#)

8.1 Configuring a Work Order Database Configuration

The following diagram shows how a work order database configuration could be set up, using the example of a work order for installing a new extension.

Figure 8-1 Graphical Representation of Work Order Database Configuration



The following flowcharts show the process that could be followed for a work order database configuration. **Figure 8-2** is a flowchart of how the configuration could work for an Install work order with the DoItNow flag set to true, and **Figure 8-3** is a flowchart showing how the configuration could work for an Install work order with the DoItNow flag not set.

Figure 8-2 Flowchart of Work Order Database Configuration with the DoItNow flag Set to True

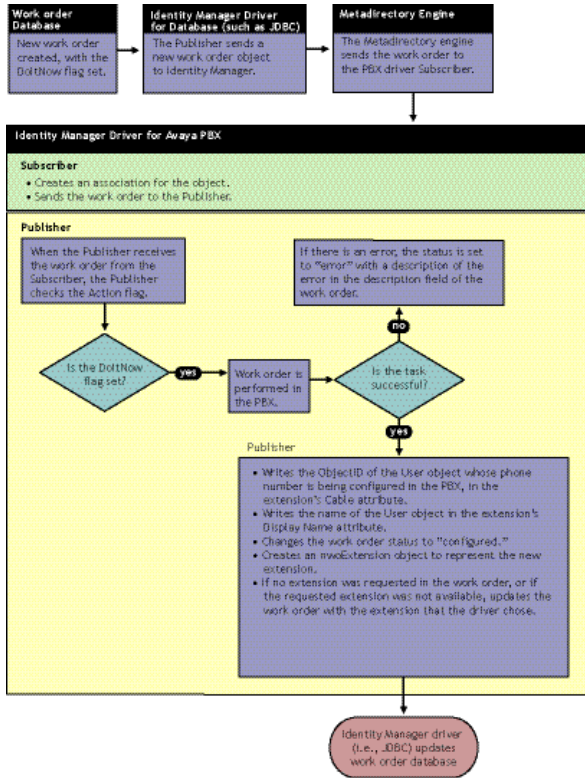
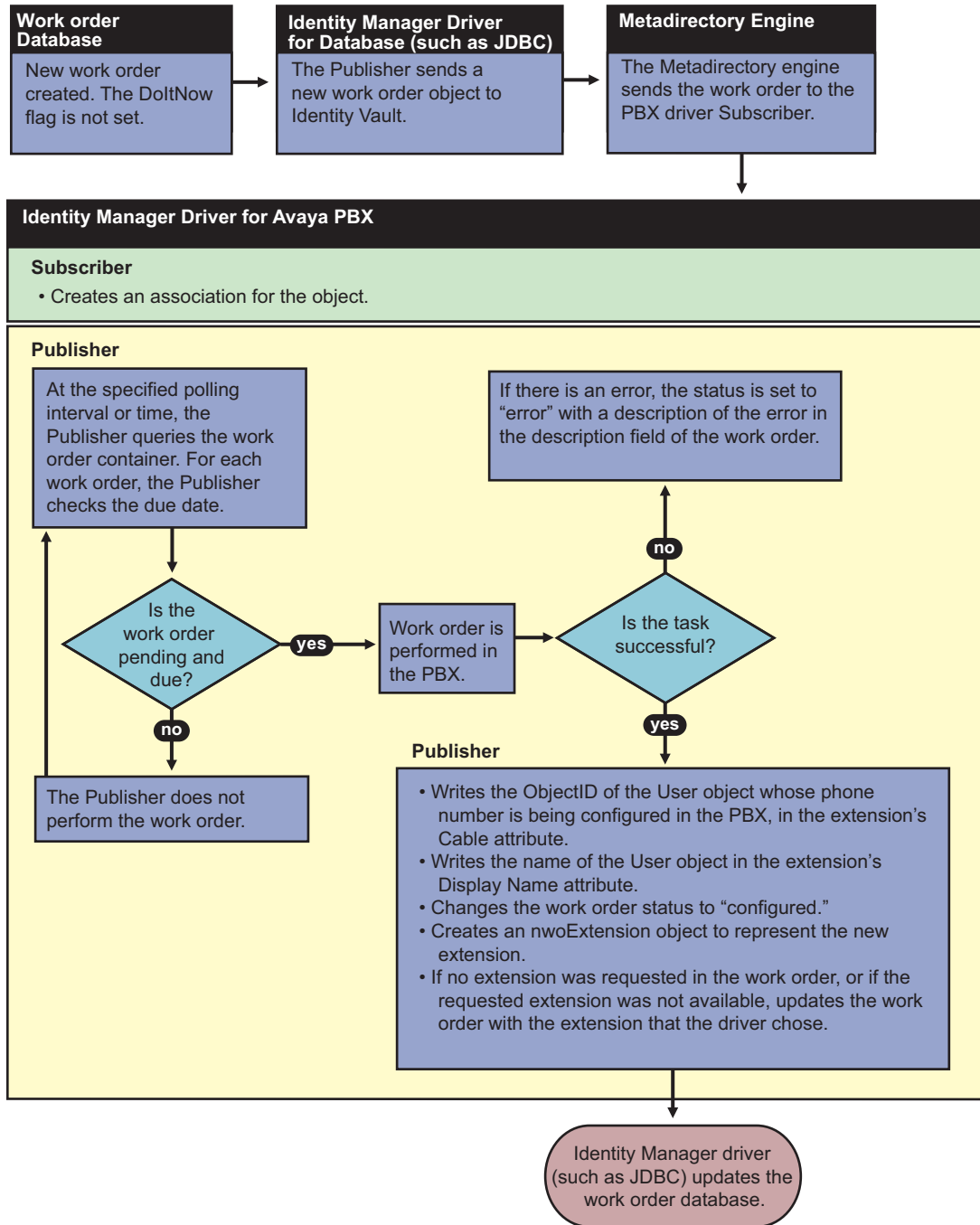


Figure 8-3 Flowchart of Work Order Database Configuration with the DoItNow Flag Not Set



8.1.1 How To Configure the Subscriber Channel

In a work order database configuration, the Subscriber channel listens for work order events.

Work orders are entered in the work order database, and are mirrored in Identity Vault as `nwoWorkOrder` objects because the JDBC driver sends them to Identity Vault. You might need to perform some data manipulation to ensure that the database work order maps correctly to the PBX driver work order.

The Identity Manager Driver for Avaya PBX performs the work orders and the results are returned to Identity Vault. The JDBC driver then updates the work order database with the results. This allows the person who entered the work order to check on the status in the work order database.

8.1.2 How To Configure the Publisher Channel

In this kind of configuration, work orders should always have the Send to Publisher Flag set to False, because all work orders are already created in the work order container. The Publisher should only update status for work orders. In contrast to the Workforce Tree configuration, the Publisher should not need to create any new work order objects.

8.2 About Work Order Systems

Many environments currently use a work order system to keep track of changes to PBX extensions. A work order database design for the driver can work well with existing work order systems. Most work order systems are forms-based front ends to a set of database tables, capable of enforcing the user's business rules and approval process. To avoid customizing the driver for each work order system, a JDBC driver or other custom driver can be responsible for synchronizing work order data between the corporate work order system and the generic work order container in the Identity Vault used by the driver.

8.3 Planning for the Work Order Database Configuration

See [Chapter 2, “Planning,”](#) on page 25.

8.4 Setting Up the Work Order Database Configuration

Follow the instructions in [Chapter 3, “Installation,”](#) on page 29, and then create a custom configuration. A file named `OracleWorkOrderNew.xml` is provided that contains sample scripts that could be helpful. However, it does not contain a complete driver configuration.

8.5 Using the Log/Reporting Functionality to Report Warnings

You might want to record the warnings that occur if a work order that is created does not match the Create policy criteria. This would mean that someone is not following the business processes.

You can keep track of this by using the log file or Novell® Audit to capture the warning messages that are generated.

Managing the Driver

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

- ♦ [Section 9.1, “Starting, Stopping, or Restarting the Driver,” on page 63](#)
- ♦ [Section 9.2, “Migrating and Resynchronizing Data,” on page 64](#)
- ♦ [Section 9.3, “Using the DirXML Command Line Utility,” on page 64](#)
- ♦ [Section 9.4, “Viewing Driver Versioning Information,” on page 65](#)
- ♦ [Section 9.5, “Reassociating a Driver Set Object with a Server Object,” on page 69](#)
- ♦ [Section 9.6, “Changing the Driver Configuration,” on page 69](#)
- ♦ [Section 9.7, “Storing Driver Passwords Securely with Named Passwords,” on page 70](#)
- ♦ [Section 9.8, “Adding a Driver Heartbeat,” on page 76](#)

9.1 Starting, Stopping, or Restarting the Driver

- ♦ [Section 9.1.1, “Starting the Driver in Designer,” on page 63](#)
- ♦ [Section 9.1.2, “Starting the Driver in iManager,” on page 63](#)
- ♦ [Section 9.1.3, “Stopping the Driver in Designer,” on page 63](#)
- ♦ [Section 9.1.4, “Stopping the Driver in iManager,” on page 63](#)
- ♦ [Section 9.1.5, “Restarting the Driver in Designer,” on page 64](#)
- ♦ [Section 9.1.6, “Restarting the Driver in iManager,” on page 64](#)

9.1.1 Starting the Driver in Designer

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

9.1.2 Starting the Driver in iManager

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

9.1.3 Stopping the Driver in Designer

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

9.1.4 Stopping the Driver in iManager

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

- 2 Browse to the driver set where the driver exists, then click *Search*.
- 3 Click the upper right corner of the driver icon, then click *Stop driver*.

9.1.5 Restarting the Driver in Designer

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

9.1.6 Restarting the Driver in iManager

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

9.2 Migrating and Resynchronizing Data

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

- ♦ **Migrate Data from Identity Vault:** Allows you to select containers or objects you want to migrate from the Identity Vault to an application. When you migrate an object, the Identity Manager engine applies all of the Matching, Placement, and Create policies, as well as the Subscriber filter, to the object.
- ♦ **Migrate Data into Identity Vault:** Assumes that the remote application (usually a Web Service) can be queried for entries that match the criteria in the publisher filter. However, because of the general nature of the Avaya 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 Avaya driver.
- ♦ **Synchronize:** The Identity Manager engine looks in the Subscriber class filter and processes all objects for those classes. Associated objects are merged. Unassociated objects are processed as Add events.

To use one of the options explained above:

- 1 In iManager, click *Identity Manager > Identity Manager Overview*.
- 2 Browse to and select the driver set where the driver exists, then click *Search*.
- 3 Click the driver icon.
- 4 Click the appropriate migration button.

For more information, see [Chapter 11, “Synchronizing Objects,” on page 83](#).

9.3 Using the DirXML Command Line Utility

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

For example, you could create a shell script on Linux to check the status of the driver. See [Appendix B, “DirXML Command Line Utility,” on page 113](#) for detailed information about the DirXML Command Line utility. For daily tasks, use iManager or Designer.

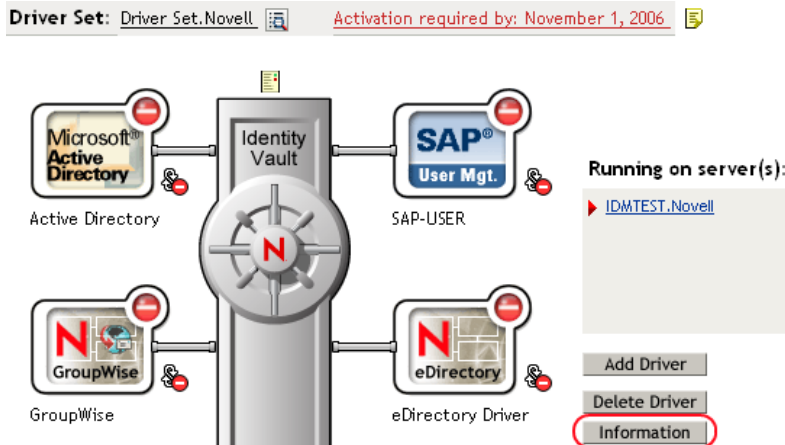
9.4 Viewing Driver Versioning Information

The Versioning Discovery tool only exists in iManager.

- ♦ [Section 9.4.1, “Viewing a Hierarchical Display of Versioning Information,” on page 65](#)
- ♦ [Section 9.4.2, “Viewing the Versioning Information As a Text File,” on page 66](#)
- ♦ [Section 9.4.3, “Saving Versioning Information,” on page 68](#)

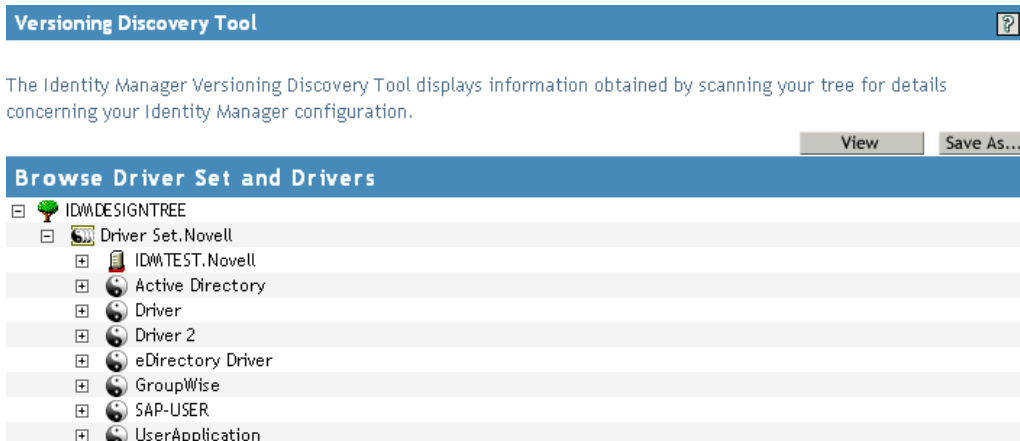
9.4.1 Viewing a Hierarchical Display of Versioning Information

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



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

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



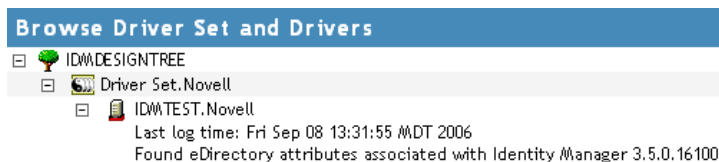
The unexpanded hierarchical view displays the following:

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

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

- ◆ Drivers

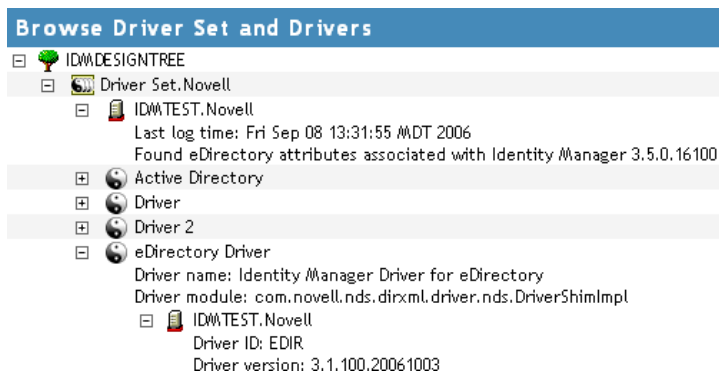
4 View versioning information related to servers by expanding the server icon.



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

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

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



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

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

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

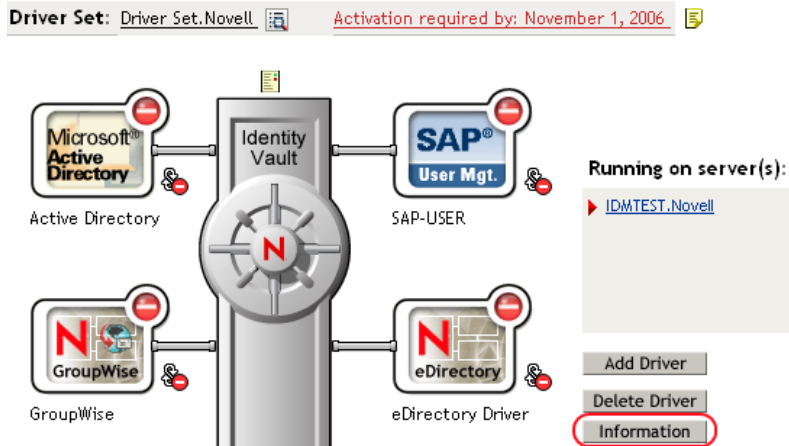
- ◆ The driver ID
- ◆ The version of the instance of the driver running on that server

9.4.2 Viewing the Versioning Information As a Text File

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

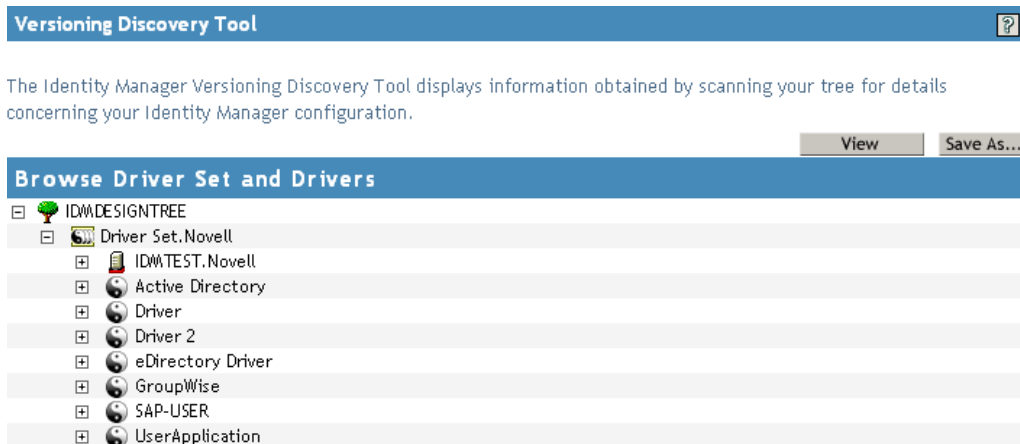
- 1 To find your Driver Set object in iManager, click *Identity Manager > Identity Manager Overview*, then click *Search*.

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



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

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



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

```

Identity Manager Version Discovery Tool v2.0
Novell, Inc. Copyright 2003, 2004

Version Query started Saturday, January 20, 2007 11:02:52 AM MST

Parameter Summary:
  Default server's DN:  IDMTEST.Novell
  Default server's IP address:  137.65.151.208
  Logged in as admin, context Novell
  Tree name:  IDMDESIGNTREE
  Found 7 Identity Manager Drivers

Driver Set:  Driver Set.Novell
  Driver Set running on Identity Vault:  IDMTEST.Novell
    Last log time:  Fri Sep 08 13:31:55 MDT 2006
    Found eDirectory attributes associated with Identity Manager 3.5.0.1
  Driver:  Active Directory.Driver Set.Novell
    Driver name:  Identity Manager Driver for Active Directory and Excha
    Driver module:  addriver.dll
    Driver Set running on Identity Vault:  IDMTEST.Novell
      Didn't find any DirXML-DriverVersion attributes associated w
      This may mean the Metadirectory engine is older than
      It does not indicate anything about the version of t
  Driver:  Driver.Driver Set.Novell
    Driver name:  Identity Manager Driver for Peoplesoft
    Driver module:  NPSShim.dll
    Driver Set running on Identity Vault:  IDMTEST.Novell

```

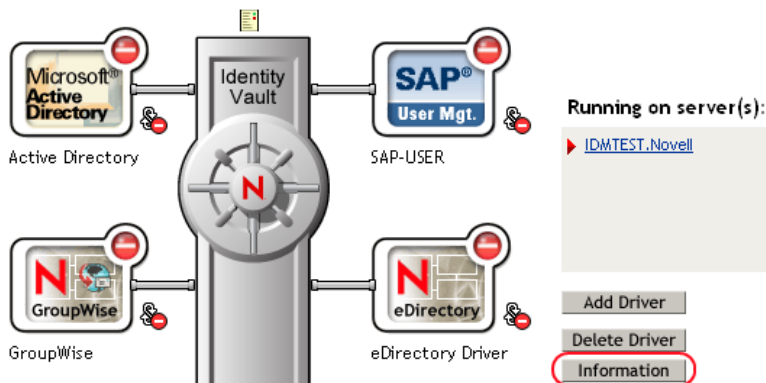
OK

9.4.3 Saving Versioning Information

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

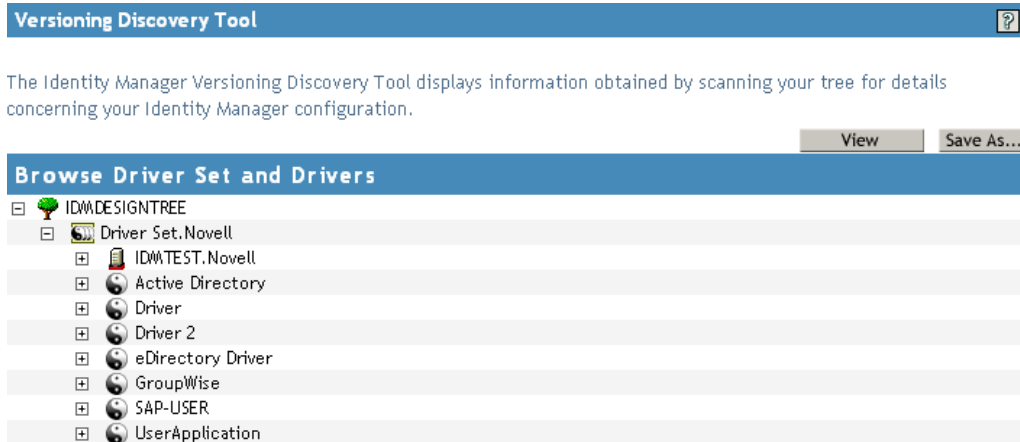
- 1 To find the Driver Set object in iManager, click *Identity Manager > Identity Manager Overview*, then click *Search*.
- 2 In the Identity Manager Overview, click *Information*.

Driver Set: Driver Set.Novell  [Activation required by: November 1, 2006](#) 



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

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



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

Identity Manager saves the data to a text file.

9.5 Reassociating a Driver Set Object with a Server Object

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

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

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

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

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

9.6 Changing the Driver Configuration

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

To change the driver configuration in iManager:

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

To change the driver configuration in Designer:

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

For a listing of all of the configuration fields, see [Appendix C, “Properties of the Driver,” on page 127](#).

9.7 Storing Driver Passwords Securely with Named Passwords

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

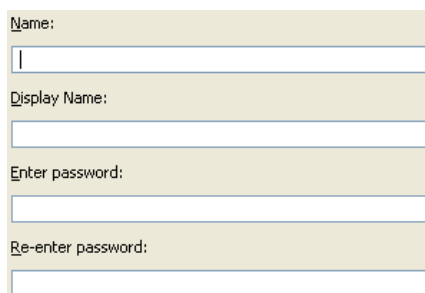
You can also use the Named Passwords feature to store other pieces of information securely, such as a user name.

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

- ♦ [Section 9.7.1, “Using Designer to Configure Named Passwords,” on page 70](#)
- ♦ [Section 9.7.2, “Using iManager to Configure Named Passwords,” on page 71](#)
- ♦ [Section 9.7.3, “Using Named Passwords in Driver Policies,” on page 72](#)
- ♦ [Section 9.7.4, “Using the DirXML Command Line Utility to Configure Named Passwords,” on page 73](#)

9.7.1 Using Designer to Configure Named Passwords

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



Name:

Display Name:

Enter password:

Re-enter password:

- 3 Specify the *Name* of the Named Password.

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

9.7.2 Using iManager to Configure Named Passwords

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

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



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

Named Password

Named Passwords lets you securely store multiple passwords for a driver. Instead of including a password in clear text in a driver policy, you can configure the policy to request a Named Password.

Name:

Display name:

Enter password:

Reenter password:

OK

Cancel

- 5 Specify a name, display name and a password, then click *OK* twice.
You can use this feature to store other kinds of information securely, such as a username.
- 6 Click *OK* to restart the driver and have the changes take effect.
- 7 To remove a Named Password, select the password name, then click *Remove*.
The password is removed without prompting you to confirm the action.

9.7.3 Using Named Passwords in Driver Policies

- ♦ [“Using the Policy Builder” on page 72](#)
- ♦ [“Using XSLT” on page 73](#)

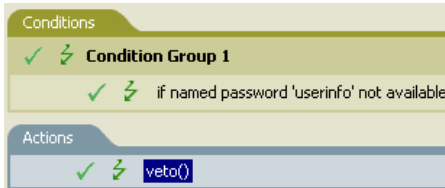
Using the Policy Builder

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

- 1 In Designer, launch Policy Builder, right-click, then click *New > Rule*.
- 2 Specify the name of the rule, then click *Next*.
- 3 Select the condition structure, then click *Next*.
- 4 Select *named password* for the *Condition*.
- 5 Browse to and select the Named Password that is stored on the driver.
In this example, it is *userinfo*.
- 6 Select whether the Operator is available or not available.
- 7 Select an action for the *Do* field.
In this example, the action is *veto*.

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

Figure 9-1 A Policy Using Named Passwords



Using XSLT

The following example shows how a named password can be referenced in a driver policy on the Subscriber channel in XSLT:

```
<xsl:value-of
select="query:getNamedPassword($srcQueryProcessor, 'mynamedpassword') "
xmlns:query="http://www.novell.com/java/
com.novell.nds.dirxml.driver.XdsQueryProcessor/>
```

9.7.4 Using the DirXML Command Line Utility to Configure Named Passwords

- “Creating a Named Password in the DirXML Command Line Utility” on page 73
- “Using the DirXML Command Line Utility to Remove a Named Password” on page 74

Creating a Named Password in the DirXML Command Line Utility

- 1 Run the DirXML Command Line utility.

For information, see [Appendix B, “DirXML Command Line Utility,”](#) on page 113.

- 2 Enter your username and password.

The following list of options appears.

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

- 3 Enter 3 for driver operations.

A numbered list of drivers appears.

- 4 Enter the number for the driver you want to add a named password to.

The following list of options appears.

```
Select a driver operation for:
driver_name
1: Start driver
2: Stop driver
```

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

- 5** Enter 13 for password operations.

The following list of options appears.

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

- 6** Enter 5 to set a new Named Password.

The following prompt appears:

```
Enter password name:
```

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

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

```
Enter password:
```

The characters you type for the password are not displayed.

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

```
Confirm password:
```

- 10** After you enter and confirm the password, you are returned to the password operations menu.
- 11** After completing this procedure, you can use the 99 option twice to exit the menu and quit the DirXML Command Line Utility.

Using the DirXML Command Line Utility to Remove a Named Password

This option is useful if you no longer need named passwords that you previously created.

- 1** Run the DirXML Command Line utility.

For information, see [Appendix B, “DirXML Command Line Utility,”](#) on page 113.

- 2** Enter your username and password.

The following list of options appears.

DirXML commands

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

3 Enter 3 for driver operations.

A numbered list of drivers appears.

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

The following list of options appears.

Select a driver operation for:

driver_name

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

5 Enter 13 for password operations.

The following list of options appears.

Select a password operation

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

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

The list of existing Named Passwords is displayed.

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

7 Enter 6 to remove one or more Named Passwords.

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

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

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

```
Enter password name:
```

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

```
Select a password operation
```

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

```
Enter choice:
```

10 (Optional) Enter 7 to see the list of existing named passwords.

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

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

9.8 Adding a Driver Heartbeat

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

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

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

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

To configure the heartbeat:

- 1** In iManager, click *Identity Manager > Identity Manager Overview*.
- 2** Browse to and select your driver set object, then click *Search*.

3 In the Identity Manager Overview, click the upper right corner of the driver icon, then click *Edit properties*.

4 On the Identity Manager tab, click *Driver Configuration*, scroll to *Driver Parameters*, then look for Heart Beat or a similar display name.

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

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

The unit of time is usually minutes; however, some drivers might choose to implement it differently, such as using seconds.

5 If a driver parameter does not exist for heartbeat, click *Edit XML*.

6 Add a driver parameter entry like the following example, as a child of <publisher-options>. (For an AD driver, make it a child of <driver-options>.)

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

TIP: If the driver does not produce a heartbeat document after being restarted, check the placement of the driver parameter in the XML.

7 Save the changes, and make sure the driver is stopped and restarted.

After you have added the driver parameter, you can edit the time interval by using the graphical view. Another option is to create a reference to a global configuration value (GCV) for the time interval. Like other global configuration values, the driver heartbeat can be set at the driver set level instead of on each individual driver object. If a driver does not have a particular global configuration value, and the driver set object does have it, the driver inherits the value from the driver set object.

Managing the Identity Manager Driver for Avaya PBX

Because the driver is meant to be customized, many aspects of managing the driver are specific to your particular implementation; this section contains some management procedures that are applicable to all implementations.

- ♦ [Section 10.1, “Changing How Often the Driver Performs Work Orders,” on page 79](#)
- ♦ [Section 10.2, “Changing the Location of PBX Site, Work Orders, User, and Extension Objects,” on page 79](#)
- ♦ [Section 10.3, “Managing Existing Users,” on page 80](#)

10.1 Changing How Often the Driver Performs Work Orders

The driver provides two ways to specify when you want work orders to be performed: polling interval and time of day. You can use either one, or both together, to achieve the frequency and timing that’s appropriate for your environment.

- 1 In iManager, go to the properties for the driver object:
 - 1a Click *DirXML Management > Overview*. Choose the driver set, and in the diagram that appears, click the *Avaya PBX driver* icon. Click it again in the next page to see the driver parameters.
- 2 To enter or change the polling interval, change the number of minutes shown.
 - ♦ If you want the driver to perform work orders at a certain polling interval, specify a number of minutes.
 - ♦ If you want to turn off the polling interval so that the driver performs work orders only at a time of day you specify, specify 0 in the field.

This choice is useful if you want to make sure that work orders are not being performed during daytime work hours.
- 3 To specify or change a certain time of day, change the time using the correct format.
 - ♦ If you want the driver to perform work orders at a certain time of day, specify a time.
 - ♦ If you want to turn off the time of day polling so that the driver performs work orders only at a polling interval, leave the field blank.

10.2 Changing the Location of PBX Site, Work Orders, User, and Extension Objects

When you import the driver from a configuration file, you are prompted to enter the location (DN) for the kinds of objects the driver needs to read:

- ♦ pbxSite
- ♦ nwoWorkOrder

- ◆ User objects
- ◆ pbxExtension objects

If you want to change the location, change the DN under the driver configuration parameters found in the driver's Properties page.

Example DN for the container holding pbxSite objects:

```
\t=CHD1_TREE\o=n\L=DirXML\O=Test\OU=PbxSite
```

Example DN for the container holding nwoWorkOrder objects:

```
\t=CHD1_TREE\o=n\L=DirXML\O=Test\OU=PbxWorkOrders
```

10.3 Managing Existing Users

After performing a work order, the driver can update a user's information with the results. For example, the driver can add a new extension to the user's list of phone numbers, or remove an extension that has been deleted, as described in [Chapter 7, "Workforce Tree Configuration," on page 49](#).

This functionality works for new and existing users, if the following conditions are met:

- ◆ The user exists in the Identity Vault.
- ◆ The policies support updating a User object after performing a work order.
- ◆ The correct ID for the user object is entered as the ObjectID in the pbxWorkOrder object.

The ObjectID in the work order is what allows the driver to update the user object when the work order is complete.

This means that your ability to manage existing users (perform work orders for their extensions and update their phone information after a work order is performed) is dependent on the accuracy with which the ObjectID is specified in the work order. This is a contrast to some other Identity Manager drivers, which require the user object to have an Identity Manager association with the driver before you can manage existing users.

Unlike some Identity Manager drivers, it is not necessary to use the `Migrate into NDS` command before being able to manage existing users. In fact, for this driver, the `Migrate into NDS` command does not affect user objects (unless you create custom policies or tools to do so).

The `Migrate into NDS` command allows you to import data from an application into Identity Vault. For the Identity Manager Driver for Avaya PBX, the `Migrate into NDS` command causes all extensions that are configured in the PBX to be created as pbxExtension objects in eDirectory™.

This action does not create associations between existing User objects in eDirectory and existing extensions. However, this list of existing extensions could be used for the following purposes as part of a manual effort or with custom tools you create:

- ◆ In the PBX, inserting the ID for the User object who uses each extension

When the driver configures an extension, it automatically enters this information in the PBX, using the ObjectID you specify in the work order. However, for existing extensions, the data might not have been filled in correctly.

This is useful if the PBX admin will continue to make changes manually to extensions, as well as using the driver to perform work orders. However, keep in mind that any manual change

must include adding the ObjectID in the PBX, or else the driver won't know which user object to update.

You can write a policy for migrating User IDs to the PBX based on their extension numbers. You can also migrate all User objects from eDirectory. To do this, set up a migration policy that creates work orders that modify ObjectID's based on extension numbers. When the migration policy has finished, change the policy to something you want for normal operations.

- ◆ Checking the accuracy of existing phone information listed for users in Identity Vault

Although the driver can add, change, or remove phone extensions listed for an existing user object after it performs a work order, it does not verify that the extensions already listed for the user are correct. So, if an incorrect extension is entered manually, that data entry error remains even after you start using the driver.

If you want to make sure that existing phone information listed for users is correct, you can compare the pbxExtension objects with the user objects by looking at the Display Name in the pbxExtension object. If display names are created with some consistency, you might be able to use a tool to match up many of the extension objects with user objects. However, some manual work is probably still required to match all of the extensions, because of duplicate employee names (such as two people who are both named John Smith) or inconsistent format of display names that were entered manually.

- ◆ Check the PBX for unassigned extensions.

You can use the nwoExtension object to find out whether you have extensions that need to be disconnected but the physical task was never completed.

- ◆ Check the display name of extensions.

You can check to make sure display names are filled in and are following any applicable corporate standards.

- ◆ Populating eDirectory with User objects

This might be an option for you if you are fairly confident in the information that is in the PBX, and you are creating a new eDirectory tree and want to populate it with users based on the users represented in the PBX. You can do this through a policy set up as you migrate User objects to eDirectory.

Synchronizing Objects

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

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

- ♦ [Section 11.1, “What Is Synchronization?” on page 83](#)
- ♦ [Section 11.2, “When Is Synchronization Done?” on page 83](#)
- ♦ [Section 11.3, “How Does the Metadirectory Engine Decide Which Object to Synchronize?” on page 84](#)
- ♦ [Section 11.4, “How Does Synchronization Work?” on page 85](#)

11.1 What Is Synchronization?

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

- ♦ Synchronization (or merging) of attribute values of an object in the Identity Vault with the corresponding attribute values of an associated object in a connected system.
- ♦ Migration of all Identity Vault objects and classes that are included in the filter on the Subscriber channel.
- ♦ Generation of the list of objects to submit to the driver’s Subscriber channel for synchronization or migration in response to a user request (a manual synchronization).
- ♦ Generation of the list of objects to submit to the driver’s Subscriber channel for synchronization or migration in response to enabling a formerly disabled driver, or in response to a cache error.

11.2 When Is Synchronization Done?

The Metadirectory engine performs object synchronization or merging in the following circumstances:

- ♦ A `<sync>` event element is submitted on the Subscriber or Publisher channel.
- ♦ A `<sync>` event element is submitted on the Subscriber channel in the following circumstances:
 - ♦ The state of the object’s association value is set to “manual” or “migrate.” (This causes an eDirectory™ event, which in turn causes the Identity Manager caching system to queue an object synchronization command in the affected driver’s cache.)
 - ♦ An object synchronization command is read from the driver’s cache.
- ♦ A `<sync>` event element is submitted on the Publisher channel in the following circumstances:
 - ♦ A driver submits a `<sync>` event element. No known driver currently does this.

- ◆ The Metadirectory engine submits a <sync> event element for each object found as the result of a migrate-into-NDS query. These <sync> events are submitted using the Subscriber thread, but are processed using the Publisher channel filter and policies.
- ◆ An <add> event (real or synthetic) is submitted on a channel and the channel Matching policy finds a matching object in the target system.
- ◆ An <add> event with an association is submitted on the Subscriber channel. This normally occurs only in exceptional cases, such as the bulk load of objects into eDirectory with DirXML-Associations attribute values.
- ◆ An <add> event is submitted on the Publisher channel and an object is found in eDirectory that already has the association value reported with the <add> event.

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

- ◆ The user issues a manual driver synchronization request. This corresponds to the *Resync* button in the Driver Set property page in ConsoleOne[®], or to the *Synchronize* button on the iManager Identity Manager Driver Overview page.
- ◆ The Metadirectory engine encounters an error with the driver's cache and cannot recover from the cache error. The driver's cache is deleted and the engine generates object synchronization commands as detailed in [Section 11.3, "How Does the Metadirectory Engine Decide Which Object to Synchronize?," on page 84.](#)

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

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

The starting filter time is used to filter objects that have modification or creation times that are older than the starting time specified in the synchronization request.

For automatically initiated driver synchronization, the starting filter time is obtained from the time stamps of cached eDirectory events. In particular, the starting filter time is the earliest time for the cached events that haven't yet been successfully processed by the driver's Subscriber channel.

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

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

1. It finds all objects that:
 - ◆ Have an entry modification time stamp greater than or equal to the starting filter time and
 - ◆ Exist in the filter on the Subscriber channel.

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

11.4 How Does Synchronization Work?

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

1. Each system (the Identity Vault and the connected system) is queried for all attribute values in the appropriate filters.
 - ♦ eDirectory is queried for all values in the Subscriber filter, and for values that are marked for synchronization in Identity Manager 2.x and Identity Manager 3.x.
 - ♦ The connected system is queried for all values in the Publisher filter, and for values that are marked for synchronization in Identity Manager 2.x and Identity Manager 3.x.
2. The returned attribute values are compared and modification lists are prepared for the Identity Vault and the connected system according to [Table 11-1 on page 86](#), [Table 11-2 on page 87](#), and [Table 11-3 on page 88](#).

In the tables the following pseudo-equations are used:

- ♦ “Left = Right” indicates that the left side receives all values from the right side.
- ♦ “Left = Right[1]” indicates that the left side receives one value from the right side. If there is more than one value, it is indeterminate.
- ♦ “Left += Right” indicates that the left side adds the right side values to the left side’s existing values.
- ♦ “Left = Left + Right” indicates that the left sides receives the union of the values of the left and right sides.

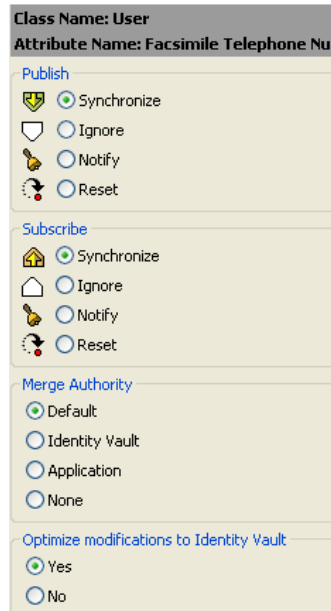
There are three different combinations of selected items in the filter, and each one creates a different output.

- ♦ [Section 11.4.1, “Scenario One,” on page 85](#)
- ♦ [Section 11.4.2, “Scenario Two,” on page 87](#)
- ♦ [Section 11.4.3, “Scenario Three,” on page 88](#)

11.4.1 Scenario One

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

Figure 11-1 Scenario One



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

Table 11-1 Output of Scenario One

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

11.4.2 Scenario Two

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

Figure 11-2 Scenario Two

Class Name: User

Attribute Name: Description

Publish

Synchronize

Ignore

Notify

Reset

Subscribe

Synchronize

Ignore

Notify

Reset

Merge Authority

Default

Identity Vault

Application

None

Optimize modifications to Identity Vault

Yes

No

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

Table 11-2 Output of Scenario Two

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

11.4.3 Scenario Three

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

Figure 11-3 Scenario Three

Class Name: User
Attribute Name: DirXML-ADAliasName

Publish

Synchronize
 Ignore
 Notify
 Reset

Subscribe

Synchronize
 Ignore
 Notify
 Reset

Merge Authority

Default
 Identity Vault
 Application
 None

Optimize modifications to Identity Vault

Yes
 No

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

Table 11-3 Output of Scenario Three

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

Backing Up the Driver

You can use Designer or iManager to create an XML file of the driver. The file contains all of the information entered into the driver during configuration. If the driver becomes corrupted, the exported file can be imported to restore the configuration information.

IMPORTANT: If the driver has been deleted, all of the associations on the objects are purged. When the XML file is imported again, new associations are created through the migration process.

Not all server-specific information stored on the driver is contained in the XML file. Make sure this information is documented through the Document Generation process in Designer. See “[Generating a Document](#)” in the *Designer 2.1 for Identity Manager 3.5.1*.

- ♦ [Section 12.1, “Exporting the Driver in Designer,” on page 89](#)
- ♦ [Section 12.2, “Exporting the Driver in iManager,” on page 89](#)

12.1 Exporting the Driver in Designer

- 1 Open a project in Designer, then right-click the driver object.
- 2 Select *Export to Configuration File*.
- 3 Specify a unique name for the configuration file, browse to location where it should be saved, then click *Save*.
- 4 Click *OK* in the Export Configuration Results window.

12.2 Exporting the Driver in iManager

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

Security: Best Practices

13

In order to secure the driver and the information it is synchronizing, see “[Security: Best Practices](#)” in the *Novell Identity Manager 3.5.1 Administration Guide*.

This section contains potential problems and error codes you might encounter while configuring or using the driver.

14.1 Troubleshooting Driver Processes

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

14.1.1 Viewing Driver Processes

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

- ♦ [“Adding Trace Levels in Designer” on page 93](#)
- ♦ [“Adding Trace Levels in iManager” on page 95](#)
- ♦ [“Capturing Driver Processes to a File” on page 96](#)

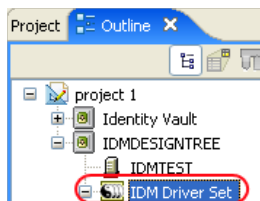
Adding Trace Levels in Designer

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

- ♦ [“Driver Set” on page 93](#)
- ♦ [“Driver” on page 94](#)

Driver Set

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



- 2 Right-click and select *Properties*, then click *5. Trace*.
- 3 Set the parameters for tracing, then click *OK*.

Parameter	Description
Driver trace level	As the driver object trace level increases, the amount of information displayed in DSTRACE increases. Trace level 1 shows errors, but not the cause of the errors. If you want to see password synchronization information, set the trace level to 5.
XSL trace level	DSTRACE displays XSL events. Set this trace level only when troubleshooting XSL style sheets. If you do not want to see XSL information, set the level to zero.
Java debug port	Allows developers to attach a Java* debugger.
Java trace file	When a value is set in this field, all Java information for the driver set object is written to a file. The value for this field is the path for that file. As long as the file is specified, Java information is written to this file. If you do not need to debug Java, leave this field blank.
Trace file size limit	Allows you to set a limit for the Java trace file. If you set the file size to <i>Unlimited</i> , the file grows in size until there is no disk space left.

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

Driver

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

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

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

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

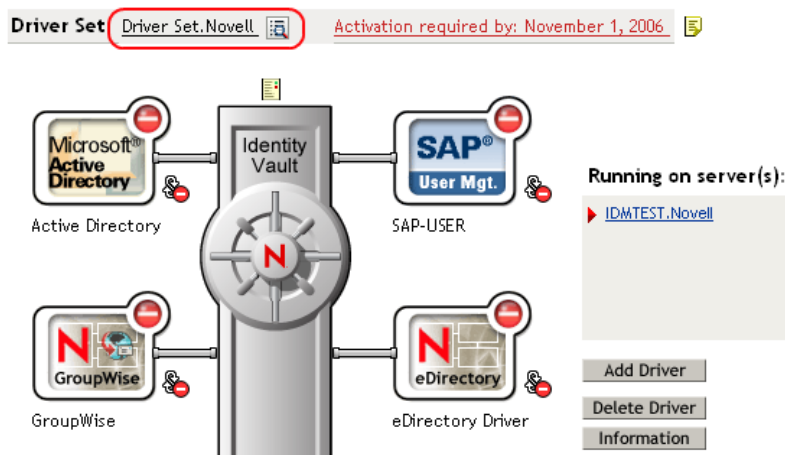
Adding Trace Levels in iManager

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

- ♦ “Driver Set” on page 95
- ♦ “Driver” on page 96

Driver Set

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



- 4 Select the *Misc* tab for the driver set object.
- 5 Set the parameters for tracing, then click *OK*.
See “Misc” on page 137 for the parameters.

Driver

- 1 In iManager, select *Identity Manager > Identity Manager Overview*.
- 2 Browse to the driver set object where the driver object resides, then click *Search*.
- 3 Click the upper right corner of the driver object, then click *Edit properties*.
- 4 Select the *Misc* tab for the driver object.
- 5 Set the parameters for tracing, then click *OK*.
See “[Misc](#)” on page 137 for the parameters.

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

Capturing Driver Processes to a File

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

The driver processes that are captured through DSTRACE are the processes that occur on the Identity Manager engine. If you use the Remote Loader, you need to capture a trace on the Remote Loader at the same time as you are capturing the trace on the Identity Manager engine.

The following methods help you capture and save Identity Manager processes through DSTRACE on different platforms.

- ♦ “[NetWare](#)” on page 96
- ♦ “[Windows](#)” on page 97
- ♦ “[UNIX](#)” on page 97
- ♦ “[iMonitor](#)” on page 97
- ♦ “[Remote Loader](#)” on page 98

NetWare

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

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

- 9 Enter `dstrace file off` at the server console.

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

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

Windows

- 1 Open the *Control Panel* > *NDS Services* > `dstrace.dlm`, then click *Start* to display the NDS Server Trace utility window.
- 2 Click *Edit* > *Options*, then click *Clear All* to clear all of the default flags.
- 3 Select *DirXML* and *DirXML Drivers*.
- 4 Click OK.
- 5 Click *File* > *New*.
- 6 Specify the filename and location where you want the DSTRACE information saved, then click *Open*.
- 7 Wait for the event to occur.
- 8 Click *File* > *Close*.
This stops the information from being written to the log file.
- 9 Open the file in a text editor and search for the event or the object you modified.

UNIX

- 1 Enter `ndstrace` to start the `ndstrace` utility.
- 2 Enter `set ndstrace=nodebug` to turn off all trace flags currently set.
- 3 Enter `set ndstrace on` to display trace messages to the console.
- 4 Enter `set ndstrace file on` to capture trace messages to the `ndstrace.log` file in the directory where eDirectory is installed. By default it is `/var/nds`.
- 5 Enter `set ndstrace+=dxml` to display the Identity Manager events.
- 6 Enter `set ndstrace+=dvrs` to display the Identity Manager driver events.
- 7 Wait for the event to occur.
- 8 Enter `set ndstrace file off` to stop logging information to the file.
- 9 Enter `exit` to quite the `ndstrace` utility.
- 10 Open the file in a text editor. Search for the event or the object that was modified.

iMonitor

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

- ♦ `ndsimon.nlm` runs on NetWare®.
 - ♦ `ndsimon.dlm` runs on Windows.
 - ♦ `ndsimonitor` runs on UNIX*.
- 1 Access iMonitor from `http://server_ip:8008/nds`.

Port 8008 is the default.

- 2 Specify a username and password with administrative rights, then click *Login*.
- 3 Select *Trace Configuration* on the left side.
- 4 Click *Clear All*.
- 5 Select *DirXML* and *DirXML Drivers*.
- 6 Click *Trace On*.
- 7 Select *Trace History* on the left side.
- 8 Click the document with the *Modification Time* of *Current* to see a live trace.
- 9 Change the *Refresh Interval* if you want to see information more often.
- 10 Select *Trace Configuration* on the left side, then click *Trace Off* to turn the tracing off.
- 11 Select *Trace History* to view the trace history.

The files are distinguished by their time stamp.

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

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

Remote Loader

You can capture the events that occur on the machine running the Remote Loader service.

- 1 Launch the Remote Loader Console by clicking the icon.
- 2 Select the driver instance, then click *Edit*.
- 3 Set the *Trace Level* to 3 or above.
- 4 Specify a location and file for the trace file.
- 5 Specify the amount of disk space that the file is allowed.
- 6 Click *OK*, twice to save the changes.

You can also enable tracing from the command line by using the following switches. For more information, see “[Configuring the Remote Loader](#)” in the *Novell Identity Manager 3.5.1 Administration Guide*.

Table 14-1 *Command Line Tracing Switches*





Option	Short Name	Parameter	Description
-trace	-t	integer	Specifies the trace level. This is only used when hosting an application shim. Trace levels correspond to those used on the Identity Manager server.
Example: <code>-trace 3</code> or <code>-t3</code>			

Option	Short Name	Parameter	Description
-tracefile	-tf	filename	<p>Specify a file to write trace messages to. Trace messages are written to the file if the trace level is greater than zero. Trace messages are written to the file even if the trace window is not open.</p> <p>Example: <code>-tracefile c:\temp\trace.txt</code> or <code>-tf c:\temp\trace.txt</code></p>
-tracefilemax	-tfm	size	<p>Specifies the approximate maximum size that trace file data can occupy on disk. If you specify this option, there is a trace file with the name specified using the tracefile option and up to 9 additional “roll-over” files. The roll-over files are named using the base of the main trace filename plus “_n”, where n is 1 through 9.</p> <p>The size parameter is the number of bytes. Specify the size by using the suffixes K, M, or G for kilobytes, megabytes, or gigabytes.</p> <p>If the trace file data is larger than the specified maximum when the Remote Loader is started, the trace file data remains larger than the specified maximum until roll-over is completed through all 10 files.</p> <p>Example: <code>-tracefilemax 1000M</code> or <code>-tfm 1000M</code></p>

Schema for PBX Management

A

As part of the installation for the Identity Manager Driver for Avaya PBX, the eDirectory™ schema is extended to include four new object classes:

- ◆  **DirXML-pbxSite**
- ◆  **DirXML-nwoWorkOrder**
- ◆  **DirXML-pbxExtension**
- ◆  **DirXML-pbxAudixSubscriber**

These objects allow the driver to connect to the PBX correctly, perform work orders, and create pbxExtension objects to represent new extension.

Installing Identity Manager 3 provides iManager plug-ins to help you create or view these objects in the Avaya PBX driver.

A.1 pbxSite Object

This object is used to represent the PBX site. The driver uses the information about the PBX site to connect to the PBX.

The driver cannot perform work orders unless the following conditions for pbxSite objects are met:

- ◆ A DirXML-pbxSite object is created for each PBX.
- ◆ The object contains correct values for the required attributes.
Not all attributes are required for all environments; some of them depend on factors such as whether you have hot or cold jacks, and whether you have non-DID phone numbers. These are noted in the tables below.
- ◆ In the driver parameters, the correct container is specified for PBX site objects.
- ◆ Each time you create or make changes to a DirXML-pbxSite object, you must stop and restart the driver.

This step is necessary so the driver recognizes the changes. The driver queries for PBX site information only at startup. Because of this, an Identity Manager association is not usually necessary for DirXML-pbxSite objects and they do not need to be included in the Filter.

The following table shows the attributes you need to specify for the PBX site. Use the iManager plug-ins to make this easier.

You must specify values for each attribute unless the table indicates that a value is not required or is conditional.

Table A-1 *pbxSite Object Attributes*

pbxSite Attributes (eDirectory Namespace)	PbxSite Attributes (Driver Namespace)	Description	Format	Sample Value
Common Name	No mapping is necessary for this in the driver namespace	This is the naming attribute for eDirectory. In the sample configurations discussed in this manual, the value comes from the PbxName attribute.	Case ignore string	ProvoPBX
DirXML-pbxName	PbxName	Local name for the PBX.	Single value case ignore string	ProvoPBX
DirXML-pbxLoginName	LoginName	Name used to authenticate to PBX.	Single value case ignore string The Identity Vault and the driver ignore the case, but the PBX itself might be case sensitive.	mylogin
DirXML-pbxPassword	Password	Password used to authenticate to PBX.	Single value case ignore string the Identity Vault and the driver ignore the case, but the PBX itself might be case sensitive.	mypassword
DirXML-pbxBrand	Brand	(Optional) Brand/ Model of PBX.	Single value case ignore string	Avaya
DirXML-pbxHotJacks	HotJacks	Are jacks hot? If your environment has cold jacks, you must also fill in the Nodes attribute.	Single value Boolean	true
DirXML-pbxAccessType	AccessType	How to access PBX: telnet. (All valid values are listed in the Sample column.)	Single value case ignore string	telnet or Emulate Audix or AudixEmulate
DirXML-pbxExtensionLength	ExtenLength	The number of digits used for extensions.	Single value numeric string	5 - 9

pbxSite Attributes (eDirectory Namespace)	PbxSite Attributes (Driver Namespace)	Description	Format	Sample Value
DirXML-pbxPhoneBlockBegin	PhoneBlockBegin	Beginning number block of DID extensions assigned to PBX.	Single value Numeric string	51000
DirXML-pbxPhoneBlockEnd	PhoneBlockEnd	Ending number block of DID extensions assigned to PBX.	Single value Numeric string	51999
DirXML-pbxNonDidPhoneBlockBegin	NonDidPhoneBlockBegin	(Conditional: This attribute is required only if you use non-DID phone numbers.) Beginning number block of non-DID extensions assigned to PBX.	Single value Numeric string	600000
DirXML-pbxNonDidPhoneBlockEnd	NonDidPhoneBlockEnd	(Conditional: This attribute is required only if you use non-DID phone numbers.) Ending number block of non-DID extensions assigned to PBX.	Single value Numeric string	60999
DirXML-pbxIpAddress	IpAddress	Telnet IP address	Single value case ignore string	133.65.121.98
DirXML-pbxNodes	Nodes	(Required only if you have cold jacks.) For example, if you had two nodes with three cabinets each, the nodes listed in this attribute might have the names shown in the sample value.	Multi value case ignore string	Building H East, 01, 02, 03 Building H West, 04, 05, 06

A.2 DirXML-nwoWorkOrder Object

The DirXML-nwoWorkOrder object (sometimes referred to as the work order object in this guide) is used to tell the driver what tasks to perform in the PBX and to allow the driver to record the results.

If your configuration is intended to update phone information for user objects after PBX tasks are complete, the ObjectID attribute must be filled in correctly on every work order. For example, in the workforce tree configuration explained in [Chapter 7, “Workforce Tree Configuration,” on page 49](#), the ObjectID is used to find and update the User object in the Identity Vault, which can subsequently be used to update phone information for the employee in the human resources application. The

ObjectID allows this flow of information to take place. Without it, the driver can perform the work order in the PBX, but no user information is updated afterward.

The Avaya driver has two flags to initiate a work order event.

DoItNow Flag

When this flag is set to True for a work order, the Subscriber wakes up the Publisher by sending the work order to the Publisher. The Publisher performs the task immediately instead of waiting for the next polling time or polling interval.

This flag is useful in a situation where you want the work order to be completed right away. You can set this flag to True when you manually create a work order, or in an automated solution you can use policies to determine whether the flag should be set. For example, you could configure the driver to set the DoItNow flag to True for an incoming work order if a corresponding attribute is set in a work order database. As another example, you could configure the driver to set the DoItNow flag to True if an Install work order is triggered by a new user in a human resources application.

SendToPublisher Flag

When this flag is set to True for a work order, the Subscriber sends the work order to the Publisher, and the Publisher writes the work order object in the correct container according to the work order container specified in the Configuration Parameters.

This flag is not necessary in implementations where the work order object is created in the Identity Vault by an administrator (as in [Chapter 5, “Base Configuration,” on page 41](#)) or an application (like the solution described in [Chapter 8, “Work Order Database Configuration,” on page 57](#)).

This flag is useful when the work order is triggered by another Identity Vault event through the use of a policy or style sheet, and the work order does not yet exist in the Identity Vault as an object. This kind of scenario is described in [Chapter 7, “Workforce Tree Configuration,” on page 49](#).

The following table shows the attributes you need to specify:

Table A-2 *nwoWorkOrder Object Attributes*

nwoWorkOrder Attributes (eDirectory Namespace)	PbxWorkOrder Attributes (Driver Namespace)	Description	Format	Sample Value
Common Name	No mapping is necessary for this in the driver namespace	The naming attribute for eDirectory. In the sample configurations discussed in this guide, the value comes from the WorkOrderNumber attribute.	Case ignore string	WorkOrder1

nwoWorkOrder Attributes (eDirectory Namespace)	PbxWorkOrder Attributes (Driver Namespace)	Description	Format	Sample Value
DirXML-nwoExtension	Extension	<p>The extension for which the work order is being performed.</p> <p>If you are installing a new extension, you can request a particular extension by entering a number here. If the extension is not available, the driver assigns the next available extension in numeric order.</p> <p>If this attribute is blank for an Install work order, the driver assigns the next available extension in numeric order.</p>	Numeric string	12345 or no value
DirXML-nwoDueDate	DueDate	<p>Work order due date.</p> <p>Must be in the format MM/DD/YYYY.</p>	Case ignore string	06/12/2003
displayName	UserName	<p>Name to be displayed on caller ID.</p> <p>You can automate the creation of this attribute based on the name of the User object.</p> <p>You can use a work order to change just the display name for an extension, if desired.</p>	Case ignore string	Doe, John
DirXML-nwoObjectID	ObjectID	<p>(Required by the policies only) The ID used by policies to associate to the user object.</p> <p>You can use an ID of your choice, such as the employee ID.</p>	Case ignore string	0804F
DirXML-pbxName	PbxName	<p>Local PBX name.</p> <p>The name of the pbxSite object you created in the Identity Vault. Use the same name as you put in the Site Object DirXML-pbxPbxName attribute.</p>	Case ignore string	ProvoPBX

nwoWorkOrder Attributes (eDirectory Namespace)	PbxWorkOrder Attributes (Driver Namespace)	Description	Format	Sample Value
DirXML-nwoNode	PbxNode	Node where the new extension should be placed. Used only for cold jacks.	Integer	Building H East Building H West
DirXML-nwoPhoneType	PhoneType	Phone type recognized by the PBX. Used by the policies to set up the extension.	Case ignore string	8410
DirXML-nwoDoltNowFlag	DoltNowFlag	When set, do the action now and ignore the due date. This work order is also sent to the Publisher.	Boolean	false
DirXML-nwoStatus	Status	Status of the work order: pending, configured, resolved, canceled, error.	Case ignore string	pending
DirXML-nwoWorkOrderNumber	WorkOrderNumber	(Optional) Unique work order number assigned by a corporate work order system other than the Identity Vault, such as a work order database.	Integer	00001
DirXML-nwoAction	Action	Work order activity: install, setcor, modify (you can modify display name or objectID), move, remove, disable, disconnect. For Audix: add, change, remove.	Case ignore string	install
Description	Description	(Optional) Description of the work order. If the status is error, this contains a description of the error.	Case ignore string	New extension
DirXML-nwoExtensionType	ExtensionType	Type of extension to configure, such as DID or non-DID.	Case ignore string	DID
DirXML-nwoJack	Jack	The jack number. Use this to help link a user ID in the Identity Vault with an extension on the PBX.	Case ignoring string	12345
DirXML-nwoSendToPublisher	SendToPublisher	If set, this work order is sent to the Publisher for further action.	Case ignore string	false

nwoWorkOrder Attributes (eDirectory Namespace)	PbxWorkOrder Attributes (Driver Namespace)	Description	Format	Sample Value
DirXML-nwoDupExtension	DupExtension	Extension to use to duplicate attributes for new configured extension.	Numeric string	19876
DirXML-nwoPort	Port	Required if you have cold jacks. Port the extension is wired to. If the port is set, the driver uses that port, whether it is set up as hot or cold jacks. If you have hot jacks, this information is not necessary; you can leave the attribute blank.	Case ignore string	10C0611
DirXML-nwoRestrictionClasses	RestrictionClass	(Optional) Class of restriction used to grant specific calling access to the extension. This is used only on setcor. If no class of restriction is specified, the extension receives the default class of restriction as defined in the PBX.	Integer	01
DirXML-nwoRoom	Room	The room number.	Case ignore string	12345

The following table shows the pbxWorkOrder association.

Table A-3 *pbxWorkOrder Association*

Driver	State	Association
AvayaPBXDriver	Associated	/User'sCommonName/Date/Time

The following table shows the required attributes for a work order that you do need to specify:

Table A-4 *Required Work Order Attributes That Must Be Specified*

Required Attributes	Description	Values or examples
DirXML-pbxName	Name of the PBX on which to perform this work order	ProvoPBX

Required Attributes	Description	Values or examples
DirXML-nwoStatus	State of the work order so the driver knows what to do with this work order	pending error warning configured
DirXML-nwoDoltNowFlag	When to perform the action on this work order	True or False
DirXML-nwoAction	What action needs to be performed on this work order	install move disable enable disconnect modify setcor

A.3 DirXML-pbxExtension Object

Table A-5 describes the attributes of the DirXML-nwoExtension object.

In the base configuration, this object is specified in the Filter and in the Schema Mapping rule.

However, in some configurations, this object is used only as output from the driver and is immediately transformed into another object by the policies, so this kind of object might never be created in the Identity Vault. For example, nwoExtension objects are not created in the workforce tree configuration explained in [Chapter 7, “Workforce Tree Configuration,” on page 49](#). Instead, nwoExtension object information that comes from the driver shim is transformed into user object information by the policies. In configurations that never create nwoExtension objects in the Identity Vault, it is not necessary to include this kind of object in the Filter and the Schema Mapping rule.

Table A-5 *pbxExtension Object Attributes*

pbxExtension Attributes (eDirectory Namespace)	pbxExtension Attributes (Driver Namespace)	Description	Format	Sample Value
Common Name	No mapping is necessary for this in the driver namespace	The naming attribute for eDirectory. In the sample configurations discussed in this guide, the value comes from the Extension attribute.	Case ignore string	12345
displayName	Name	The Display name of the extension this object represents.	Case ignore string	Doe, John
DirXML-nwoExtension	Extension	The extension number.	Case ignore string	12345

pbxExtension Attributes (eDirectory Namespace)	pbxExtension Attributes (Driver Namespace)	Description	Format	Sample Value
DirXML-nwoJack	Jack	The jack number. Use this to help link a user ID in the Identity Vault to an extension on the PBX.	Case ignore string	12345
DirXML-nwoRoom	Room	The room name or number. Use this to help link a user ID in the Identity Vault to an extension on the PBX.	Case ignore string	up to 15 characters: Monitorroom
Description	Description	Description of object.	Case ignore string	Extension configured.
DirXML-nwoObjectID	ObjectID	Entry ID of the object (such as a user object) that this phone number belongs to. In the sample configurations this is the entry ID, but it could be an ID of your choice, such as the employee ID.	Case ignore string	0000804F
DirXML-nwoPhoneType	PhoneType	Phone type recognized by the PBX.	Case ignore string	8410
DirXML-nwoRestrictionClass	RestrictionClass	Class of restriction, used to grant specific calling access to the extension. For example, you might define class 01 with a restriction that no international calls can be made.	Integer	01
(Port) DirXML-nwoPort	Port	Required if you have cold jacks. Port the extension is wired to. If the port is set, the driver uses that port, whether it is set up for hot or cold jacks. If you have hot jacks, this information is not necessary; you can leave the attribute blank.	Case ignore string	10C0611
DirXML-pbxName	PbxName	Local name for the PBX.	Single value case ignore string	ProvoPBX

The following table shows the association for the pbxExtension object.

Table A-6 *pbxExtension Object Association*

Driver	State	Association
AvayaPBXDriver	Associated	<i>/Driver Name/Extension</i> For example, <i>/Avaya PBX Driver/12345</i>

A.4 DirXML-pbxAudixSubscriber Object

Table A-7 describes the attributes of the DirXML-pbxAudixSubscriber object.

Adding Audix support to the current Avaya driver requires the driver to access and manipulate PBX subscriber objects. This is performed by extending the eDirectory schema to include a DirXML-pbxAudixSubscriber object. For Audix support, the driver uses `add`, `change` and `remove` work order commands. Other important information includes:

- ◆ The Audix server is separate from the extension server, so you must specify a new site object for the Audix server.
- ◆ The Audix server supports different commands, so the Avaya driver supports an Audix type of access. The new DirXML-AccessType is `audix` and the new actions supported are `add`, `change`, `remove` and `AudixEmulate`.
- ◆ The Avaya driver does not synchronize manual changes made on the Audix server to the Identity Vault.
- ◆ The Avaya driver makes no attempt to see if there is a corresponding extension on an `add` command. The creation of extension and subscriber objects are independent of one another and can be done in any order.
- ◆ Because the Audix server is separate and very different than the extension server, separate work order commands are provided for Audix support. Any successful actions and errors resulting from work orders are reported back through the work order. Work order commands are `add`, `change`, and `remove`.
 - ◆ On an `add` command, the driver checks to see if the Subscriber Extension object is on the the PBX Audix server. If the object already exists, the driver generates an error and reports it back in the work order.
 - ◆ On a `change` command, the driver checks to see if the Subscriber object already exists. If it does not, the driver generates an error and reports it back through the work order.
 - ◆ On a `delete` command, the driver checks to see if the subscriber object is already there. If it is not, the driver generates an error and reports it in the work order.

Table A-7 *pbxAudixSubscriber Object Attributes*

pbxAudixSubscriber Attributes (eDirectory Namespace)	pbxAudixSubscriber (Driver Namespace)	Description	Format	Sample Value
Class Name	No mapping is necessary for this in the driver namespace	The naming attribute for the Identity Vault. In the sample configurations discussed in this guide, the value comes from the Audix Subscriber attribute.	Case ignore string	12345
displayName	Name	The Display name of the extension this object represents.	Case ignore string	Doe, John
DirXML-nwoExtension	Extension	The extension number.	Case ignore string	12345
DirXML-pbxName	PbxName	Local name for the PBX.	Single value case ignore string	ProvoPBX

A.5 User Objects and Their Identity Manager Associations

The driver can be configured to create work orders based on creation of or changes to a User object, as discussed in [Chapter 7, “Workforce Tree Configuration,” on page 49](#).

Two existing attributes of a User object are used in the workforce tree configuration. Because these attributes already exist in the eDirectory schema, the schema extension that is part of the driver installation does not make any changes to User objects.

User objects receives an Identity Manager association for the driver when they are sent to the Subscriber channel. The table below shows the value of the association for a User object.

Table A-8 *User Object Association*

Driver	State	Association
AvayaPBXDriver	Associated	<i>/PBXName/WorkOrderCommonName/Time</i>

You might expect that the association for a user object for the Avaya PBX Driver is the phone extension. However, this is not the case. The association for the user object is made before a work order is performed. For a work order to install a new extension, the number of the new phone extension is not known until the work order is performed, so the extension can’t be used.

DirXML Command Line Utility

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

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

eDirectory 8.7.x

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

eDirectory 8.8.x

- ♦ Windows: \Novell\Nds\dxcmd.bat
- ♦ NetWare[®]: sys:\system\dxcmd.ncf
- ♦ UNIX: /opt/novell/eDirectory/bin/dxcmd

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

- ♦ [Section B.1, “Interactive Mode,” on page 113](#)
- ♦ [Section B.2, “Command Line Mode,” on page 122](#)

B.1 Interactive Mode

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

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

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

- 4 Enter the number of the command you want to perform.
[Table B-1 on page 114](#) contains the list of options and what functionality is available.
- 5 Enter 99 to quit the utility.

NOTE: If you are running eDirectory™ 8.8 on UNIX or Linux*, you must specify the -host and -port parameters. For example, `dxcmd -host 10.0.0.1 -port 524`. If the parameters are not specified, a jclient error occurs.

`novell.jclient.JCException: connect (to address) 111 UNKNOWN ERROR`

By default, eDirectory 8.8 is not listening to localhost. The DirXML Command Line utility needs to resolve the server IP address or hostname and the port to be able to authenticate.

Table B-1 *Interactive Mode Options*

Option	Description
1: <i>Start Driver</i>	Starts the driver. If there is more than one driver, each driver is listed with a number. Enter the number of the driver to start the driver.
2: <i>Stop Driver</i>	Stops the driver. If there is more than one driver, each driver is listed with a number. Enter the number of the driver to stop the driver.
3: <i>Driver operations</i>	Lists the operations available for the driver. If there is more than one driver, each driver is listed with a number. Enter the number of the driver to see the operations available. See Table B-2 on page 115 for a list of operations.
4: <i>Driver set operations</i>	Lists the operations available for the driver set. <ul style="list-style-type: none">◆ 1: Associate driver set with server◆ 2: Disassociate driver set from server◆ 99: Exit
5: <i>Log events operations</i>	Lists the operations available for logging events through Novell® Audit. See Table B-5 on page 120 for a description of these options.
6: <i>Get DirXML version</i>	Lists the version of the Identity Manager installed.
7: <i>Job operations</i>	Manages jobs created for Identity Manager.1
99: <i>Quit</i>	Exits the DirXML Command Line utility

Figure B-1 *Driver Options*

```
Select a driver operation for:
Active Directory.Driver Set.Novell.IDMDESIGNTREE.

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

Enter choice:
```

Table B-2 *Driver Options*

Options	Description
1: <i>Start driver</i>	Starts the driver.
2: <i>Stop driver</i>	Stops the driver.
3: <i>Get driver state</i>	Lists the state of the driver. <ul style="list-style-type: none">◆ 0 - Driver is stopped◆ 1 - Driver is starting◆ 2 - Driver is running◆ 3 - Driver is stopping
4: <i>Get driver start option</i>	Lists the current driver start option. <ul style="list-style-type: none">◆ 1 - Disabled◆ 2 - Manual◆ 3 - Auto
5: <i>Set driver start option</i>	Changes the start option of the driver. <ul style="list-style-type: none">◆ 1 - Disabled◆ 2 - Manual◆ 3 - Auto◆ 99 - Exit

Options	Description
6: <i>Resync driver</i>	<p>Forces a resynchronization of the driver. It prompts for a time delay: <i>Do you want to specify a minimum time for resync? (yes/no)</i>.</p> <p>If you enter Yes, specify the date and time you want the resynchronization to occur: <i>Enter a date/time (format 9/27/05 3:27 PM)</i>.</p> <p>If you enter No, the resynchronization occurs immediately.</p>
7: <i>Migrate from application into DirXML</i>	<p>Processes an XML document that contains a query command: <i>Enter filename of XDS query document:</i></p> <p>Create the XML document that contains a query command by using the Novell nds.dtd (http://developer.novell.com/ndk/doc/dirxml/dirxmlbk/ref/ndsstd/query.html).</p> <p>Examples:</p> <p>NetWare: <code>sys:\files\query.xml</code></p> <p>Windows: <code>c:\files\query.xml</code></p> <p>Linux: <code>/files/query.xml</code></p>
8: <i>Submit XDS command document to driver</i>	<p>Processes an XDS command document:</p> <p><i>Enter filename of XDS command document:</i></p> <p>Examples:</p> <p>NetWare: <code>sys:\files\user.xml</code></p> <p>Windows: <code>c:\files\user.xml</code></p> <p>Linux: <code>/files/user.xml</code></p> <p><i>Enter name of file for response:</i></p> <p>Examples:</p> <p>NetWare: <code>sys:\files\user.log</code></p> <p>Windows: <code>c:\files\user.log</code></p> <p>Linux: <code>/files/user.log</code></p>
9: <i>Submit XDS event document to driver</i>	<p>Processes an XDS event document:</p> <p><i>Enter filename of XDS event document:</i></p> <p>Examples:</p> <p>NetWare: <code>sys:\files\add.xml</code></p> <p>Windows: <code>c:\files\add.xml</code></p> <p>Linux: <code>/files/add.xml</code></p>

Options	Description
10: <i>Queue event for driver</i>	Adds and event to the driver queue <i>Enter filename of XDS event document:</i> Examples: NetWare: <code>sys:\files\add.xml</code> Windows: <code>c:\files\add.xml</code> Linux: <code>/files/add.xml</code>
11: <i>Check object password</i>	Validates that an object's password in the connected system is associated with a driver. It matches the object's eDirectory password (Distribution Password, used with Universal Password). <i>Enter user name:</i>
12: <i>Initialize new driver object</i>	Performs an internal initialization of data on a new Driver object. This is only for testing purposes.
13: <i>Password operations</i>	There are nine Password options. See Table B-3 on page 117 for a description of these options.
14: <i>Cache operations</i>	There are five Cache operations. See Table B-4 on page 119 for a descriptions of these options.
99: <i>Exit</i>	Exits the driver options.

Figure B-2 Password Operations

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

Table B-3 Password Operations

Operation	Description
1: <i>Set shim password</i>	Sets the application password. This is the password of the user account you are using to authenticate into the connected system with.
2: <i>Clear shim password</i>	Clears the application password.

Operation	Description
3: <i>Set Remote Loader password</i>	<p>The Remote Loader password is used to control access to the Remote Loader instance.</p> <p>Enter the Remote Loader password, then confirm the password by typing it again.</p>
4: <i>Clear Remote Loader password</i>	<p>Clears the Remote Loader password so no Remote Loader password is set on the Driver object.</p>
5: <i>Set named password</i>	<p>Allows you to store a password or other pieces of security information on the driver. See Section 9.7, "Storing Driver Passwords Securely with Named Passwords," on page 70 for more information.</p> <p>There are four prompts to fill in:</p> <ul style="list-style-type: none"> ◆ <i>Enter password name:</i> ◆ <i>Enter password description:</i> ◆ <i>Enter password:</i> ◆ <i>Confirm password:</i>
6: <i>Clear named passwords</i>	<p>Clears a specified named password or all named passwords that are stored on the driver object: <i>Do you want to clear all named passwords? (yes/no).</i></p> <p>If you enter Yes, all Named Passwords are cleared. If you enter No, you are prompted to specify the password name that you want to clear.</p>
7: <i>List named passwords</i>	<p>Lists all named passwords that are stored on the driver object. It lists the password name and the password description.</p>
8: <i>Get password state</i>	<p>Lists if a password is set for:</p> <ul style="list-style-type: none"> ◆ Driver Object password ◆ Application password ◆ Remote loader password <p>The dxcmd utility allows you to set the Application password and the Remote Loader password. You cannot set the Driver Object password with this utility. It shows if the password has been set or not.</p>
99: <i>Exit</i>	<p>Exits the current menu and takes you back to the Driver options.</p>

Figure B-3 Cache Operations

```
Enter choice: 14

Select a cache operation

1: Get driver cache limit
2: Set driver cache limit
3: View cached transactions
4: Delete cached transactions
99: Exit

Enter choice:
```

Table B-4 Cache Operations

Operation	Description
1: <i>Get driver cache limit</i>	Displays the current cache limit that is set for the driver.
2: <i>Set driver cache limit</i>	Sets the driver cache limit in kilobytes. A value of 0 is unlimited.
3: <i>View cached transactions</i>	A text file is created with the events that are stored in cache. You can select the number of transactions to view. <ul style="list-style-type: none">◆ <i>Enter option token (default=0):</i>◆ <i>Enter maximum transactions records to return (default=1):</i>◆ <i>Enter name of file for response:</i>
4: <i>Delete cached transactions</i>	Deletes the transactions stored in cache. <ul style="list-style-type: none">◆ <i>Enter position token (default=0):</i>◆ <i>Enter event-id value of first transaction record to delete (optional):</i>◆ <i>Enter number of transaction records to delete (default=1):</i>
99: <i>Exit</i>	Exits the current menu and takes you back to the Driver options.

Figure B-4 Log Event Operations

```
Select a log events operation

1: Set driver set log events
2: Reset driver set log events
3: Set driver log events
4: Reset driver log events
99: Exit

Enter choice:
```

Table B-5 *Log Events Operations*

Operation	Description
1: <i>Set driver set log events</i>	Allows you to log driver set events through Novell Audit. There are 49 items you can select to log. See Table B-6 on page 120 for a list of these options. Type the number of the item you want to log. After the items are selected, enter 99 to accept the selections.
2: <i>Reset driver set log events</i>	Resets all of the log event options.
3: <i>Set driver log events</i>	Allows you to log driver events through Novell Audit. There are 49 items to select to log. See Table B-6 on page 120 for a list of these options. Type the number of the item you want to log. After the items are selected, enter 99 to accept the selections.
4: <i>Reset driver log events</i>	Resets all of the log event options.
99: <i>Exit</i>	Exits the log events operations menu.

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

Options
1: Status success
2: Status retry
3: Status warning
4: Status error
5: Status fatal
6: Status other
7: Query elements
8: Add elements
9: Remove elements
10: Modify elements
11: Rename elements
12: Move elements
13: Add-association elements
14: Remove-association elements
15: Query-schema elements
16: Check-password elements

Options

- 17: Check-object-password elements
 - 18: Modify-password elements
 - 19: Sync elements
 - 20: Pre-transformed XDS document from shim
 - 21: Post input transformation XDS document
 - 22: Post output transformation XDS document
 - 23: Post event transformation XDS document
 - 24: Post placement transformation XDS document
 - 25: Post create transformation XDS document
 - 26: Post mapping transformation <inbound> XDS document
 - 27: Post mapping transformation <outbound> XDS document
 - 28: Post matching transformation XDS document
 - 29: Post command transformation XDS document
 - 30: Post-filtered XDS document <Publisher>
 - 31: User agent XDS command document
 - 32: Driver resync request
 - 33: Driver migrate from application
 - 34: Driver start
 - 35: Driver stop
 - 36: Password sync
 - 37: Password request
 - 38: Engine error
 - 39: Engine warning
 - 40: Add attribute
 - 41: Clear attribute
 - 42: Add value
 - 43: Remove value
 - 44: Merge entire
 - 45: Get named password
 - 46: Reset Attributes
 - 47: Add Value - Add Entry
 - 48: Set SSO Credential
-

Options

49: Clear SSO Credential

50: Set SSO Passphrase

51: User defined IDs

99: Accept checked items

Table B-7 *Enter Table Title Here*

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

B.2 Command Line Mode

The command line mode allows you to use script or batch files. [Table B-8 on page 122](#) lists the different options that are available.

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

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

This example command starts the driver.

Table B-8 *Command Line Options*

Option	Description
Configuration	
<code>-user <user name></code>	Specify the name of a user with administrative rights to the drivers you want to test.
<code>-host <name or IP address></code>	Specify the IP address of the server where the driver is installed.
<code>-password <user password></code>	Specify the password of the user specified above.

Option	Description
-port <port number>	Specify a port number, if the default port is not used.
-q <quiet mode>	Displays very little information when a command is executed.
-v <verbose mode>	Displays detailed information when a command is executed.
-s <stdout>	Writes the results of the <code>dxcmd</code> command to <code>stdout</code> .
-? <show this message>	Displays the help menu.
-help <show this message>	Displays the help menu.
Actions	
-start <driver dn>	Starts the driver.
-stop <driver dn>	Stops the driver.
-getstate <driver dn>	Shows the state of the driver as running or stopped.
-getstartoption <driver dn>	Shows the startup option of the driver.
-setstartoption <driver dn> <disabled manual auto> <resync noresync>	Sets how the driver starts if the server is rebooted. Sets whether the objects are to be resynchronized when the driver restarts.
-getcachelimit <driver dn>	Lists the cache limit set for the driver.
-setcachelimit <driver dn> <0 or positive integer>	Sets the cache limit for the driver.
-migrateapp <driver dn> <filename>	Processes an XML document that contains a query command.
	Create the XML document that contains a query command by using the Novell <code>nds.dtd</code> (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_dtd/data/dtdndsoverview.html#dtdndsoverview).
-setshimpassword <driver dn> <password>	Sets the application password. This is the password of the user account you are using to authenticate into the connected system with.
-clearshimpassword <driver dn> <password>	Clears the application password.
-setremoteloaderpassword <driver dn> <password>	Sets the Remote Loader password.
	The Remote Loader password is used to control access to the Remote Loader instance.
<clearremoteloaderpassword <driver dn>	Clears the Remote Loader password.

Option	Description
-sendcommand <driver dn> <input filename> <output filename>	<p>Processes an XDS command document.</p> <p>Specify the XDS command document as the input file.</p> <p>Examples:</p> <p>NetWare: sys:\files\user.xml</p> <p>Windows: c:\files\user.xml</p> <p>Linux: /files/user.log</p> <p>Specify the output filename to see the results.</p> <p>Examples:</p> <p>NetWare: sys:\files\user.log</p> <p>Windows: c:\files\user.log</p> <p>Linux: /files/user.log</p>
-sendevent <driver dn> <input filename>	<p>Submits a document to the driver's Subscriber channel, bypassing the driver cache. The document is processed ahead of anything that might be in the cache at the time of the submission. It also means that the submission fails if the driver is not running.</p>
-queueevent <driver dn> <input filename>	<p>Submits a document to the driver's Subscriber channel by queuing the document in the driver cache. The document gets processed after anything that might be in the cache at the time of the submission. The submission won't fail if the driver isn't running.</p>
-setlogevents <dn> <integer ...>	<p>Sets Novell Audit log events on the driver. The integer is the option of the item to log. See Table B-6 on page 120 for the list of the integers to enter.</p>
-clearlogevents <dn>	<p>Clears all Novell Audit log events that are set on the driver.</p>
-setdriverset <driver set dn>	<p>Associates a driver set with the server.</p>
-cleardriverset	<p>Clears the driver set association from the server.</p>
-getversion	<p>Shows the version of Identity Manager that is installed.</p>
-initdriver object <dn>	<p>Performs an internal initialization of data on a new Driver object. This is only for testing purposes.</p>
-setnamedpassword <driver dn> <name> <password> [description]	<p>Sets named passwords on the driver object. You specify the name, the password, and the description of the named password.</p>
-clearnamedpassword <driver dn> <name>	<p>Clears a specified named password.</p>
-startjob <job dn>	<p>Starts the specified job.</p>

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

If a command line is executed successfully, it returns a zero. If the command line returns anything other than zero, it is an error. For example 0 means success, and -641 means invalid operation. -641 is an eDirectory error code. [Table B-9 on page 125](#) contains other values for specific command line options.


Table B-9 *Command Line Option Values*

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

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

Properties of the Driver

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

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

- ◆ [Section C.1, “Driver Configuration,” on page 127](#)
- ◆ [Section C.2, “Global Configuration Values,” on page 132](#)
- ◆ [Section C.3, “Named Passwords,” on page 132](#)
- ◆ [Section C.4, “Engine Control Values,” on page 133](#)
- ◆ [Section C.5, “Log Level,” on page 135](#)
- ◆ [Section C.6, “Driver Image,” on page 136](#)
- ◆ [Section C.7, “Security Equals,” on page 136](#)
- ◆ [Section C.8, “Filter,” on page 136](#)
- ◆ [Section C.9, “Edit Filter XML,” on page 137](#)
- ◆ [Section C.10, “Misc,” on page 137](#)
- ◆ [Section C.11, “Excluded Users,” on page 138](#)
- ◆ [Section C.12, “Driver Manifest,” on page 138](#)
- ◆ [Section C.13, “Driver Inspector,” on page 139](#)
- ◆ [Section C.14, “Driver Cache Inspector,” on page 139](#)
- ◆ [Section C.15, “Inspector,” on page 140](#)
- ◆ [Section C.16, “Server Variables,” on page 140](#)

C.1 Driver Configuration

In iManager:

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

In Designer:

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

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

C.1.1 Driver Module



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

In iManager:

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

In Designer:

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

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

C.1.2 Driver Object Password

In iManager:

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

In Designer:

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

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

C.1.3 Authentication







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





In iManager:

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

In Designer:

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

Option	Description
<i>Authentication ID</i> or  <i>User ID</i>	Specify a user application ID. This ID is used to pass Identity Vault subscription information to the application. Example: Administrator
<i>Authentication Context</i> or  <i>Connection Information</i>	Specify the IP address or name of the server the application shim should communicate with.
<i>Remote Loader Connection Parameters</i> or  <i>Host name</i>  <i>Port</i>  <i>KMO</i>  <i>Other parameters</i>	Used only if the driver is connecting to the application through the remote loader. The parameter to enter is <code>hostname=xxx.xxx.xxx.xxx port=xxxx</code> <code>kmo=certificatename</code> , when the host name is the IP address of the application server running the Remote Loader server and the port is the port the remote loader is listening on. The default port for the Remote Loader is 8090. The <code>kmo</code> entry is optional. It is only used when there is an SSL connection between the Remote Loader and the Metadirectory engine. Example: <code>hostname=10.0.0.1 port=8090</code> <code>kmo=IDMCertificate</code>

Option	Description
<i>Driver Cache Limit (kilobytes)</i>	Specify the maximum event cache file size (in KB). If it is set to zero, the file size is unlimited.
or	 Click <i>Unlimited</i> to set the file size to unlimited in Designer.
 <i>Cache limit (KB)</i>	
<i>Application Password</i>	Specify the password for the user object listed in the <i>Authentication ID</i> field.
or	
 <i>Set Password</i>	
<i>Remote Loader Password</i>	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.
or	
 <i>Set Password</i>	

C.1.4 Startup Option


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

In iManager:

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

In Designer:

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

Option	Description
<i>Auto start</i>	The driver starts every time the Identity Manager server is started.
<i>Manual</i>	The driver does not start when the Identity Manager server is started. The driver must be started through Designer or iManager.
<i>Disabled</i>	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.
 <i>Do not automatically synchronize the driver</i>	This option only applies if the driver is deployed and was previously disabled. If this is not selected, the driver re-synchronizes the next time it is started.

C.1.5 Driver Parameters

In iManager:

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

In Designer:

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

Parameter	Description
Driver Options	
Driver parameters for server: Server-name	Enter the name of the server whose driver parameters you want to modify.
Subscriber Options	None specified.
Publisher Options	
<i>PBX Site Objects DN</i>	Specify the domain name of the PBX site object. This is the same name you use for the creation of the PBX site object in the directory. For example: <code>\o=Novell\l=AvayaPBX\ou=PbxSite</code>
<i>PBX Work Orders Objects DN</i>	Specify the domain name of the PBX work orders object. This is the same name you use for the creation of the PBX work orders object in the directory. For example: <code>\t=ARROW-TREE\o=Novell\l=AvayaPBX\ou=PbxWorkOrders</code>
<i>Poll Interval (minutes)</i>	Specifies the number of minutes between checks for available transactions to process. The default is 5.
<i>Poll Time</i>	Select a poll time for the driver to use. For example, 12:00 AM.
<i>IP Address for LDAP Host for Emulation</i>	For emulation, the driver needs to access an LDAP host (eDirectory or any other host). Specify the IP address of the LDAP server.
<i>Port of LDAP Host Emulation</i>	Specify the port number of the LDAP server used for emulation. The default is 389.
<i>DN of the Login User for Emulation</i>	For emulation, the driver needs to know which username has access to the LDAP server. This user must have rights to read/write to the extension containers on the server. Specify the name of the user for the LDAP server.
<i>User Password for Emulation</i>	Specify the password of the user who accesses the LDAP server. Confirm the password. You can also clear the password by selecting <i>Clear Password</i> .

Parameter	Description
<i>Extension Container DN for Emulation</i>	Specify the DN of the container where the driver will add or modify extension objects (this container must already exist on the LDAP server.) For example, ou=PbxExtensions,ou=AvayaPBX,o=Novell

C.2 Global Configuration Values

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

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

In iManager:

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

In Designer:

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

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

For the Avaya PBX driver, no global configuration values are specified.

C.3 Named Passwords

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

You can also use the Named Passwords feature to store other pieces of information securely, such as a user name. To configured Named Passwords, see [Section 9.7, “Storing Driver Passwords Securely with Named Passwords,” on page 70](#).

In iManager:

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

In Designer:

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

C.4 Engine Control Values

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

In iManager:

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

This option does not exist in Designer at this time.

Table C-1 *Engine Control Values*

Option	Description
<i>Subscriber channel retry interval in seconds</i>	The Subscriber channel retry interval controls how frequently the Metadirectory engine retries the processing of a cached transaction after the application shim's Subscriber object returns a retry status.
<i>Qualified form for DN-syntax attribute values</i>	The qualified specification for DN-syntax attribute values controls whether values for DN-syntax attribute values are presented in unqualified slash form or qualified slash form. A True setting means the values are presented in qualified form.
<i>Qualified form from rename events</i>	The qualified form for rename events controls whether the new-name portion of rename events coming from the Identity Vault are presented to the Subscriber channel with type qualifiers. For example, CN=. A True setting means the names are presented in qualified form.
<i>Maximum eDirectory replication wait time in seconds</i>	The maximum eDirectory™ replication wait time controls the maximum time that the Metadirectory engine waits for a particular change to replicate between the local replica and a remote replica. This only affects operations where the Metadirectory engine is required to contact a remote eDirectory server in the same tree to perform an operation and might need to wait until some change has replicated to or from the remote server before the operation can be completed (for example, object moves when the Identity Manager server does not hold the master replica of the moved object; file system rights operations for Users created from a template.)

Option	Description
<i>Use non-compliant backwards-compatible mode for XSLT</i>	<p>This control sets the XSLT processor used by the Metadirectory engine to a backwards-compatible mode. The backwards-compatible mode causes the XSLT processor to use one or more behaviors that are not XPath 1.0 and XSLT 1.0 standards-compliant. This is done in the interest of backwards-compatibility with existing DirXML[®] style sheets that depend on the non-standard behaviors.</p> <p>For example, the behavior of the XPath “!=” operator when one operand is a node-set and the other operand is other than a node-set is incorrect in DirXML releases up to and including Identity Manager 2.0. This behavior has been corrected; however, the corrected behavior is disabled by default through this control in favor of backwards-compatibility with existing DirXML style sheets.</p>
<i>Maximum application objects to migrate at once</i>	<p>This control is used to limit the number of application objects that the Metadirectory engine requests from an application during a single query that is performed as part of a Migrate Objects from Application operation.</p> <p>If java.lang.OutOfMemoryError errors are encountered during a Migrate from Application operation, this number should be set lower than the default. The default is 50.</p> <hr/> <p>NOTE: This control does not limit the number of application objects that can be migrated; it merely limits the batch size.</p>
<i>Set creatorsName on objects created in Identity Vault</i>	<p>This control is used by the Identity Manager engine to determine if the creatorsName attribute should be set to the DN of this driver on all objects created in the Identity Vault by this driver.</p> <p>Setting the creatorsName attribute allows for easily identifying objects created by this driver, but also carries a performance penalty. If not set, the creatorsName attribute defaults to the DN of the NCP[™] Server object that is hosting the driver.</p>
<i>Write pending associations</i>	<p>This control determines whether the Identity Manager engine writes a pending association on an object during Subscriber channel processing.</p> <p>Writing a pending association confers little or no benefit but does incur a performance penalty. Nevertheless, the option exists to turn it on for backward compatibility.</p>
<i>Use password event values</i>	<p>This control determines the source of the value reported for the nspmDistributionPassword attribute for Subscriber channel Add and Modify events.</p> <p>Setting the control to False means that the current value of the nspmDistributionPassword is obtained and reported as the value of the attribute event. This means that only the current password value is available. This is the default behavior.</p> <p>Setting the control to True means that the value recorded with the eDirectory event is decrypted and is reported as the value of the attribute event. This means that both the old password value (if it exists) and the replacement password value at the time of the event are available. This is useful for synchronizing passwords to certain applications that require the old password to enable setting a new password.</p>

Option	Description
<i>Enable password synchronization status reporting</i>	<p>This control determines whether the Identity Manager engine reports the status of Subscriber channel password change events.</p> <p>Reporting the status of Subscriber channel password change events allows applications such as the Identity Manager User Application to monitor the synchronization progress of a password change that should be synchronized to the connected application.</p>

C.5 Log Level

Every driver set and driver has a log level field where you can define the level of errors that should be tracked. The level you indicate here determines which messages are available to the logs. By default, the log level is set to track error messages (this also includes fatal messages). Change the log level if you want to track additional message types.


Novell® recommends that you use Novell Audit instead of setting the log levels. See “[Integrating Identity Manager with Novell Audit](#)” in the *Identity Manager 3.5.1 Logging and Reporting*.

In iManager:

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

In Designer:

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

Option	Description
<i>Use log settings from the DriverSet</i>	If this is selected, the driver logs events as the options are set on the Driver Set object.
<i>Log errors</i>	Logs just errors
<i>Log errors and warnings</i>	Logs errors and warnings
<i>Log specific events</i>	Logs the events that are selected. Click the  icon to see a list of the events.
<i>Only update the last log time</i>	Updates the last log time.
<i>Logging off</i>	Turns logging off for the driver.
<i>Turn off logging to DriverSet, Subscriber and Publisher logs</i>	If selected, turns all logging off for this driver on the Driver Set object, Subscriber channel, and the Publisher channel.
<i>Maximum number of entries in the log (50-500)</i>	Number of entries in the log. The default value is 50.

C.6 Driver Image

Allows you to change the image associated with the driver. You can browse and select a different image from the default image.

The image associated with a driver is used by the Identity Manager Overview plug-in when showing the graphical representation of your Identity Manager configuration. Although storing an image is optional, it makes the overview display more intuitive.

NOTE: The driver image is maintained when a driver configuration is exported.

In iManager:

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

In Designer:

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

C.7 Security Equals

Use the Security page to view or change the list of objects that the driver is explicitly security equivalent to. This object effectively has all rights of the listed objects.

If you add or delete an object in the list, the system automatically adds or deletes this object in that object's "Security Equal to Me" property. You don't need to add the [Public] trustee or the parent containers of this object to the list, because this object is already implicitly security equivalent to them.

In iManager:

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

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

C.8 Filter

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

In iManager:

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

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

- 1 In an open project, click the *Outline* tab.
- 2 Select the driver you want to manage the filter for, then click the plus sign to the left.
- 3 Double-click the *Filter* icon to launch the Filter Editor.

C.9 Edit Filter XML

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

In iManager:

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

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

- 1 In an open project, click the *Outline* tab.
- 2 Select the driver you want to manage the filter for, then click the plus sign to the left.
- 3 Double-click the *Filter* icon and to launch the Filter Editor, then click *XML Source* at the bottom of the Filter Editor.

C.10 Misc

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


In iManager:

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

In Designer:

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

Option	Description
<i>Trace level</i>	Increases the amount of information displayed in DSTRACE. Trace level 1 shows errors, but not the cause of the errors. If you want to see password synchronization information, set the trace level to 5.

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

C.11 Excluded Users

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

In iManager:

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

Designer does not list the excluded users.

C.12 Driver Manifest

The driver manifest is like a resumé for the driver. It states what the driver supports, and includes a few configuration settings. The driver manifest is created by default when the Driver object is imported. A network administrator usually does not need to edit the driver manifest.

In iManager:

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

In Designer:

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

C.13 Driver Inspector

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

- ♦ **Driver:** A link to run the *Driver Overview* on the driver that is being inspected.
- ♦ **Driver Set:** A link to run the *Driver Set Overview* of the driver set that holds the driver.
- ♦ **Delete:** Deletes the associations of the selected objects.
- ♦ **Refresh:** Select this option to re-read all of the objects associated with the driver and refresh the displayed information.
- ♦ **Actions:** Allows you to perform actions on the objects associated with the driver. Click *Actions* to expand the menu, which includes:
 - ♦ **Show All Associations:** Displays all objects associated with the driver.
 - ♦ **Filter for Disabled Associations:** Displays all objects associated with the driver that have a Disabled state.
 - ♦ **Filter for Manual Associations:** Displays all objects associated with the driver that have a Manual state.
 - ♦ **Filter for Migrate Associations:** Displays all objects associated with the driver that have a Migrate state.
 - ♦ **Filter for Pending Associations:** Displays all objects associated with the driver that have a Pending state.
 - ♦ **Filter for Processed Associations:** Displays all objects associated with the driver that have a Processed state.
 - ♦ **Filter for Undefined Associations:** Displays all objects associated with the driver that have an Undefined state.
 - ♦ **Association Summary:** Displays the state of all objects associated with the driver.
- ♦ **Object DN:** Displays the DN of the associated objects.
- ♦ **State:** Displays the association state of the object.
- ♦ **Object ID:** Displays the value of the association.

C.14 Driver Cache Inspector

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

- ♦ **Driver:** A link to run the *Driver Overview* on the driver that is associated with this cache file.
- ♦ **Driver Set:** A link to run the *Driver Set Overview* on the driver set that holds the driver.
- ♦ **Driver's cache on:** Lists the server object that contains this instance of the cache file.
- ♦ **Start/Stop Driver icons:** Displays the current state of the driver and allows you to start or stop the driver.
- ♦ **Edit icon:** Allows you to edit the properties of the currently selected Server object.
- ♦ **Delete:** Deletes the selected items from the cache file.
- ♦ **Refresh:** Select this option to re-read the cache file and refresh the displayed information.

- ♦ **Show:** Limits the number of items to be displayed. The options are:
 - ♦ 25 per page
 - ♦ 50 per page
 - ♦ 100 per page
 - ♦ Other: Allows you to specify a desired number.
- ♦ **Actions:** Allows you to perform actions on the entries in the cache file. Click *Actions* to expand the menu, which includes:
 - ♦ **Expand All:** Expands all of the entries displayed in the cache file.
 - ♦ **Collapse All:** Collapses all of the entries displayed in the cache file.
 - ♦ **Go To:** Allows you to access a specified entry in the cache file. Specify the entry number, then click *OK*.
 - ♦ **Cache Summary:** Summarizes all of the events stored in the cache file.

C.15 Inspector

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

C.16 Server Variables

This page lets you enable and disable Password Synchronization and the associated options for the selected driver.

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

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

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

To change these settings in iManager:

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

This option is available from a drop-down list or a tab, depending on your version of iManager and your browser.

- 4 Select *Configuration Options*, make changes, then click *OK*.

NOTE: Enabling or disabling options on this page corresponds to values of True or False for certain global configuration values (GCVs) used for password synchronization in the driver parameters. Novell recommends that you edit them here in the graphical interface, instead of on

the GCVs page. This interface helps ensure that you don't set conflicting values for the password synchronization GCVs.

Option	Description
<i>Identity Manager accepts password (Publisher Channel)</i>	<p>If this option is enabled, Identity Manager allows passwords to flow from the connected system driver into the Identity Vault data store.</p> <p>Disabling this option means that no <password> elements are allowed to flow to Identity Manager. They are stripped out of the XML by a password synchronization policy on the Publisher channel.</p> <p>If this option is enabled, and the option below it for Distribution Password is disabled, a <password> value coming from the connected system is written directly to the Universal Password in the Identity Vault if it is enabled for the user. If the user's password policy does not enable Universal Password, the password is written to the NDS Password.</p>
<i>Use Distribution Password for password synchronization</i>	<p>To use this setting, you must have a version of eDirectory that supports Universal Password, regardless of whether you have enabled Universal Password in your password policies.</p> <p>If this option is enabled, a password value coming from the connected system is written to the Distribution Password. The Distribution Password is reversible, which means that it can be retrieved from the Identity Vault data store for password synchronization. It is used by Identity Manager for bidirectional password synchronization with connected systems. For Identity Manager to distribute passwords to connected systems, this option must be enabled.</p> <p>NMAS and Password policies control whether the Distribution Password is synchronized with other passwords in the Identity Vault. By default, the Distribution Password is the same as the Universal Password in the Identity Vault.</p> <p>If the password in the Identity Vault is to be independent of Password Synchronization, so that Identity Manager is a conduit only for synchronizing passwords among connected systems, change this default setting. In the Universal Password Configuration Options in a Password policy, disable <i>Synchronize Universal Password with Distribution Password</i>. This use of Identity Manager Password Synchronization is also referred to as "tunneling."</p>
<i>Accept password only if it complies with user's Password Policy</i>	<p>To use this setting, users must have a Password policy assigned that has Universal Password enabled, and Advanced Password Rules enabled and configured.</p> <p>If this option is chosen, Identity Manager does not write a password from this connected system to the Distribution Password in the Identity Manager data store or publish it to connected systems unless the password complies with the user's Password policy.</p> <p>By using the notification option that is also on this page, you can inform users when a password is not set because it is not compliant.</p>

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