Novell exteNd Composer™

5.2.1

www.novell.com

INTEGRATION MODULE FOR ENTERPRISE - CUSTOM EDITION GUIDE FOR IDENTITY MANAGER

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 makes changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

This product may require export authorization from the U.S. Department of Commerce prior to exporting from the U.S. or Canada.

Title to the Software and its documentation, and patents, copyrights and all other property rights applicable thereto, shall at all times remain solely and exclusively with SilverStream and its licensors, and you shall not take any action inconsistent with such title. The Software is protected by copyright laws and international treaty provisions. You shall not remove any copyright notices or other proprietary notices from the Software or its documentation, and you must reproduce such notices on all copies or extracts of the Software or its documentation. You do not acquire any rights of ownership in the Software.

Patents pending.

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

www.novell.com

Novell exteNd Composer *Integration Module for Enterprise - Custom Edition Guide for Identity Manager*
September, 2005

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

Novell Trademarks

SilverStream Trademarks

Third-Party Trademarks

Third-Party Software Legal Notices

The Apache Software License, Version 1.1

Autonomy

Bouncy Castle

Castor Library

5. Due credit should be given to the ExoLab? Project (http://www.exolab.org/).

THIS SOFTWARE IS PROVIDED BY INTALIO AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTALIO OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Indiana University Extreme! Lab Software License

Version 1.1.1

Copyright (c) 2002 Extreme! Lab, Indiana University. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Indiana University Extreme! Lab (http://www.extreme.indiana.edu/)."

Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Indiana University" and "Indiana University Extreme! Lab" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact http://www.extreme.indiana.edu/.

5. Products derived from this software may not use "Indiana University" name nor may "Indiana University" appear in their name, without prior written permission of the Indiana University.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS, COPYRIGHT HOLDERS OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

JDOM.JAR

Copyright (C) 2000-2002 Brett McLaughlin & Jason Hunter. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the disclaimer that follows these conditions in the documentation and/or other materials provided with the distribution.

3. The name "JDOM" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact license@jdom.org.

4. Products derived from this software may not be called "JDOM", nor may "JDOM" appear in their name, without prior written permission from the JDOM Project Management (pm@jdom.org).

In addition, we request (but do not require) that you include in the end-user documentation provided with the redistribution and/or in the software itself an acknowledgement equivalent to the following: "This product includes software developed by the JDOM Project (http://www.jdom.org/)."

Alternatively, the acknowledgment may be graphical using the logos available at http://www.jdom.org/images/logos.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE JDOM AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Phaos

This Software is derived in part from the SSLavaTM Toolkit, which is Copyright ©1996-1998 by Phaos Technology Corporation. All Rights Reserved. Customer is prohibited from accessing the functionality of the Phaos software.

W3C

W3C® SOFTWARE NOTICE AND LICENSE

This work (and included software, documentation such as READMEs, or other related items) is being provided by the copyright holders under the following license. By obtaining, using and/or copying this work, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions.

Permission to copy, modify, and distribute this software and its documentation, with or without modification, for any purpose and without fee or royalty is hereby granted, provided that you include the following on ALL copies of the software and documentation or portions thereof, including modifications:

1. The full text of this NOTICE in a location viewable to users of the redistributed or derivative work.
2. Any pre-existing intellectual property disclaimers, notices, or terms and conditions. If none exist, the W3C Software Short Notice should be included (hypertext is preferred, text is permitted) within the body of any redistributed or derivative code.
3. Notice of any changes or modifications to the files, including the date changes were made. (We recommend you provide URIs to the location from which the code is derived.)

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR DOCUMENTATION.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to the software without specific, written prior permission. Title to copyright in this software and any associated documentation will at all times remain with copyright holders.

Contents

About This Book

Purpose

This guide describes how to use the Integration Module for Enterprise - Custom Edition to create Identity Manager drivers. This product has design-time as well as runtime executables and uses J2EE Connector Architecture technology to provide integration capability.

Audience

This book is for developers who are planning to use Novell exteNd Composer to develop drivers for Identity Manager.

Prerequisites

You should be familiar with the exteNd Composer work environment and deployment options. You should be familiar with the external application that you want to integrate with eDirectory, and you should be familiar with the Composer Connect that provides access to the API of the external application. You should also be familiar with eDirectory and the basics of developing Identity Manager drivers. Familiarity with Java Connector Architecture is helpful but not required.

Software Versions

The following software versions are required to use the Integration Module for Enterprise - Custom Edition:

- exteNd Composer Version 5.2.1 (Enterprise), build 1307D or higher (to check the Composer build level, select Help > About Composer; then select the System button). You can find exteNd Composer patches at the following URL: http://support.novell.com/filefinder/15595/index.html.
- eDirectory 8.7.3 or higher
- Identity Manager 2 or higher

Additional Documentation

For the complete set of Novell exteNd documentation, see the Novell Documentation Web site (http://www.novell.com/documentation-index/index.jsp).

1 Introduction

Welcome to the *Integration Module for Enterprise - Custom Edition*. This Guide is a companion to the *Novell exteNd Composer User's Guide*, which details how to use all the features of exteNd Composer except for the Connect Component Editors. Novell exteNd Composer™ provides separate Component Editors for each Connect. The special features of each Connect are described in separate Guides. You should be familiar with the *Novell exteNd Composer User's Guide* and the Guide for the Connect that you will be using before using this Guide.

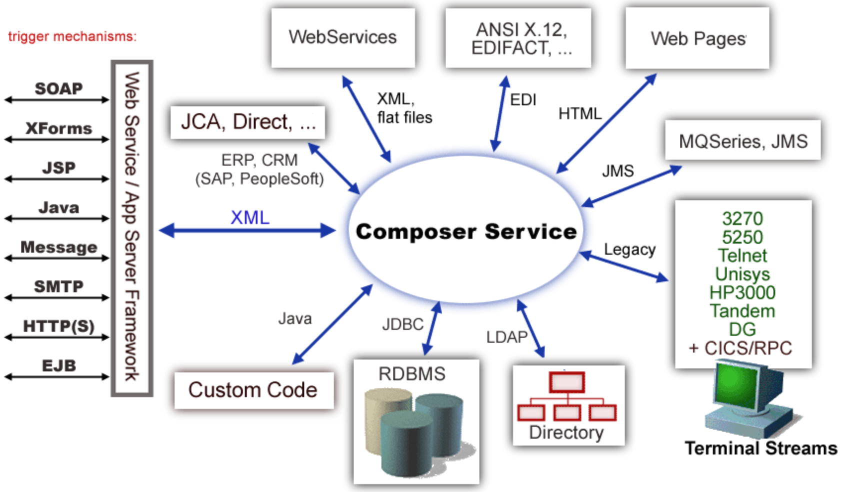
NOTE: To be successful with this Integration Module for Enterprise - Custom Edition, you must be familiar with the external application system that you are integrating with Identity Manager, and with the Composer Connect for that application system.

About Novell exteNd Composer™

Novell exteNd Composer is the XML integration-broker component of the Novell exteNd suite. It encompasses a set of design tools for building XML integration applications and Web services, plus a runtime engine that enables execution and administration of the services that you build. The applications and services that you build with exteNd Composer can be deployed to any popular J2EE application server or servlet container. Supported application servers include JBoss, IBM WebSphere and BEA WebLogic in addition to the Novell exteNd application server. Apache Jakarta Tomcat is also supported. Check the Novell web site for latest platform-support information.

At the core of exteNd Composer is a robust XML transformation engine capable of performing a wide range of data transformations, including joining of multiple documents, decomposition of documents, and creation of entirely new documents. The underlying enabling technologies include XSLT, XPath, ECMAScript, and Java. The exteNd Composer design environment offers a rich, intuitive graphical user interface, making it possible for you to specify XML transformations and mappings visually, using wizards, dialogs, and drag-and-drop gestures. You never have to write raw XSL or Java code.

exteNd Composer supports numerous kinds of data-source connectivity, through individual adapters called Connects. Using the functionality exposed in the various Connects, you can design enterprise application integration (EAI) applications and Web services that pull data in from or push data out to different kinds of back-end systems, using a variety of transport protocols and technologies, ranging from 3270 and 5250 terminal data streams to Telnet, HP3000, Unisys T27 (and UTS), Tandem, and Data General, in addition to HTML screen-scraping, JMS messaging, and CICS RPC transactions. You can also take advantage of JDBC, LDAP, and other mechanisms to reach back-end data repositories and systems that might or might not natively understand XML. Composer Connects allow you to connect to these systems inobtrusively, so as to marshall non-XML data into XML form or vice versa without any need to modify host-system setups or code.



In addition to legacy data-stream and protocol-specific Connects, exteNd Composer has Connects for ERP and CRM systems, including Baan, PeopleSoft, SAP, Lawson, J.D. Edwards, Siebel, and Oracle Applications. As with other Connect solutions, the ERP and CRM Connects are fully integrated into exteNd Composer's design-time environment so that you can use intuitive visual tools to create powerful custom integration solutions, eliminating the need to write Java code or edit raw XML or schemas by hand. You can also test the components that you build against live connections, using the animation facility (step-through debugger) of the design environment. As part of the design and debug process, your application-aware components can call other exteNd Composer Components (such as XML Map Components, JDBC Components, etc.) and make use of any of the core actions that exteNd Composer defines (such as Map, Function, Log, and other actions).

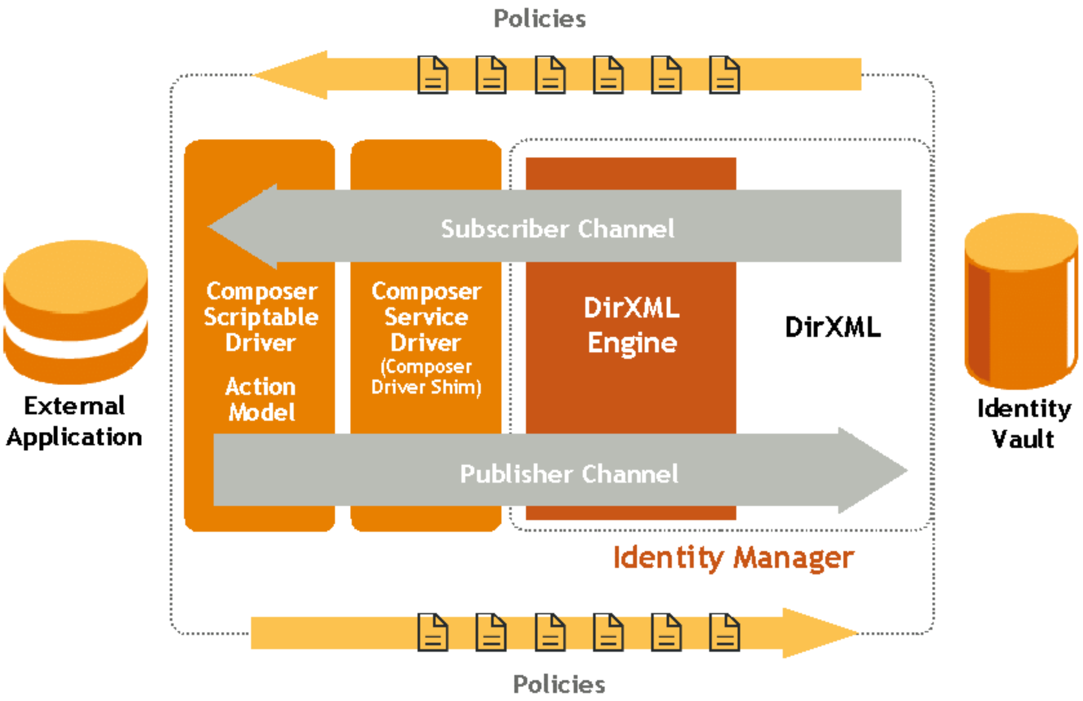
About the exteNd Composer Integration Module for Enterprise - Custom Edition

The exteNd Composer Integration Module for Enterprise - Custom Edition consists of the following major components:

- Composer Service Driver (ComposerDriverShim.jar, Composer runtime files, and a configuration file), which acts as a bridge between Identity Manager and exteNd Composer. The Composer Service Driver is installed automatically to eDirectory when you deploy a driver from exteNd Composer. It is also included in Identity Manager 3.
- A collection of template xObjects that can be added to any new exteNd Composer project when the project is created. The template xObjects demonstrate how to create Web services that support the Identity Manager commands and events specified in the NDS.dtd, and that detect changes in the external application and publish the changes to Identity Manager.

Using the Integration Module for Enterprise - Custom Edition, you can write exteNd Composer Action models that integrate with external applications. Action models connect to external applications through a Composer Connect and perform creates, deletes, modifies, and queries as specified by the Identity Manager API.

Connector logic in exteNd Composer is encapsulated in a component that contains an Action model. An Action model is composed of generalized programming constructs and connector-specific actions. The component accepts XML as its input, performs general and connector-specific actions, and returns XML as its output. This enables applications written in exteNd Composer to call external applications using a pure XML interface. In a Subscriber model, the connector-specific actions transform the XML input into a format usable by the external application. The connector-specific action then interprets this format and makes native API calls to the target system as a request, accepts the response from the external application, and returns the response back to the component in XML format. This XML can then be processed further or used as the output of the component. In a Publisher model, a target-specific exteNd Composer listener component receives XML from the XML API, applies any business logic, and essentially mirrors the Subscriber process.



An Action model can make multiple API calls and apply any type of data mappings, transformations, constraints, rules, and policies to the data before or after communications with the target host. exteNd Composer then generates the XML documents defining the information needed for the functions of the external application.

At runtime, the Composer Connect receives each Composer-generated XML document and creates the actual function call to the API of the external application, using the information contained in the received XML documents. The Composer Connect contains no application integration logic. Application logic is implemented in the Action model, not the connector. This requires the exteNd Composer developer to have a detailed understanding of the APIs available in the connected system and how those APIs are used to accomplish the desired task.

Which External Applications are Supported?

The exteNd Composer Integration Module for Enterprise - Custom Edition provides a framework for developing Identity Manager drivers. You can use this framework for developing drivers for any application system for which there is a corresponding Composer Connect (see your exteNd Composer documentation for a complete list of Composer Connects). All Composer Connects provide Subscriber channel support. In addition, exteNd Composer provides Publisher channel support in the following Connects:

- Baan
- Lawson
- J.D. Edwards
- JMS
- Oracle Applications
- PeopleSoft
- SAP
- Siebel

What Kinds of Applications Can You Build Using the Integration Module for Enterprise - Custom Edition?

You can use the Integration Module for Enterprise - Custom Edition to build custom applications that connect disparate directories using eDirectory and allow data to be automatically shared and synchronized across directories. The drivers can connect directories despite different naming conventions and schema definitions. You can build any kind of Web service or integration application that needs to push data into or pull data from eDirectory using XML as the interchange format.

Where Should I Start?

If you are new to Identity Manager driver development, Novell DeveloperNet® University provides a free, online Identity Manager driver creation training course. Although the course teaches how to develop drivers using C++ or Java (while exteNd Composer uses an Action model developed in Composer's high level language), it provides an excellent background to Identity Manager driver development.

DeveloperNet University: Custom Driver Development Course:
http://developer.novell.com/education/tutorials/dirxml2/01.htm

The Novell Identity Manager Driver Kit (NDK) also includes documentation that provides background information about Identity Manager driver development:

http://developer.novell.com/ndk/dirxml.htm

2 Getting Started

Enabling the Integration Module for Enterprise - Custom Edition Feature

The Integration Module for Enterprise - Custom Edition feature is not enabled in exteNd Composer by default. You must enable the feature by editing the exteNd Composer configuration file, xconfig.xml.

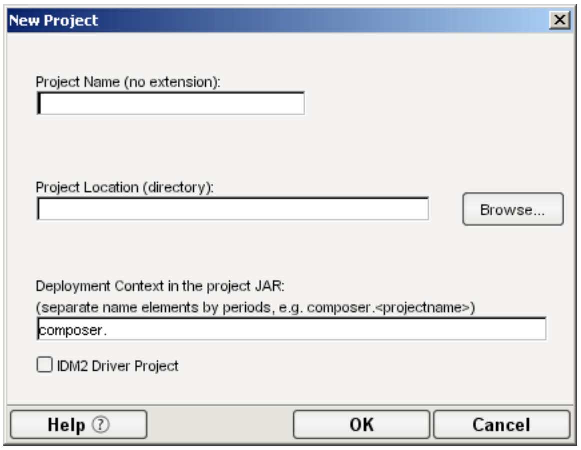
➢ **To enable the exteNd Composer Integration Module for Enterprise - Custom Edition feature:**

1. If you haven't already done so, install exteNd Composer (see the *Installing Novell exteNd* manual).
2. If exteNd Composer is running, exit from Composer before performing the following steps.
3. Locate your **xconfig.xml** file in the **/Composer/Designer/bin** directory and open the file in a text editor.
4. Locate the element named "IDM2_WIZARD ENABLED".
5. Change the value of this element to TRUE (IDM2_WIZARD ENABLED="TRUE").
6. Save your changes.

Creating an Integration Module for Enterprise - Custom Edition Project

➢ **To create a Integration Module for Enterprise - Custom Edition Project:**

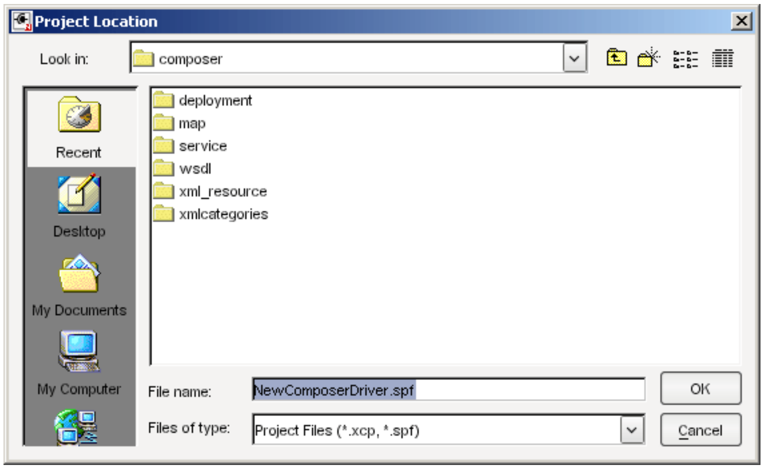
1. Start exteNd Composer.
2. Select **New Project** from the File menu. The **New Project** dialog box is displayed:


New Project
Project Name (no extension):
Project Location (directory):
Browse...
Deployment Context in the project JAR:
(separate name elements by periods, e.g. composer.<projectname>)
composer.
☐ IDM2 Driver Project
Help ⓘ
OK
Cancel

3. Select the **IDM2 Driver Project** option (located in the lower left of the dialog box). This is the option that enables the Integration Module for Enterprise - Custom Edition feature.
4. Type in a **Project Name**. This is a required field. Composer adds the project name extension (.spf).

5 Select **Browse** to locate the directory in which you want your project to reside. The **Project Location** dialog box is displayed.

NOTE: If you have a project open, the **Project Location** dialog box shows the directory where the open project resides.



File name: NewComposerDriver.spf

Files of type: Project Files (*.xcp, *.spf)

6 Navigate to the directory in which you want your project to reside. If necessary, create a new directory for the project.

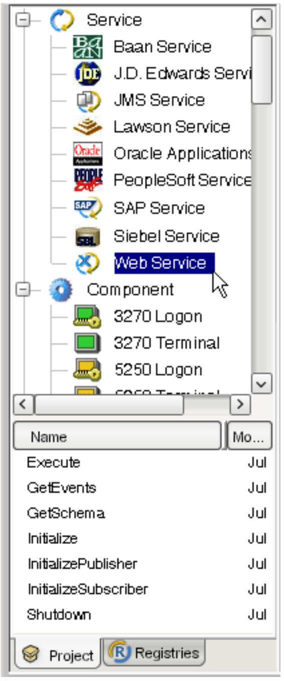
NOTE: The **File Name** in the Project Location dialog defaults to the project name you designated in Step 4.

7 Click **OK**. The **New Project** dialog appears with the **Project Name** and **Project Location** displayed.

8 Enter a deployment context string in the **Deployment Context in the project JAR** field. The string should contain labels (no spaces) separated by periods, as in “com.server.apps.”

NOTE: The context string should not contain Java-language keywords, such as *try, catch, finally, int,* or *for.* For a complete list of Java keywords, see the “Reserved Words” appendix in the *Novell exteNd Composer User’s Guide.*

9 Click **OK**. The Composer window appears with the name of the project that you just created in the title bar. You can view the template artifacts that are included in the project by using the **Project** tab in the Navigator Frame. For example, to view the Web services that are included in the project, scroll down to the Services section of the Navigator Frame, then double-click on Web Services:



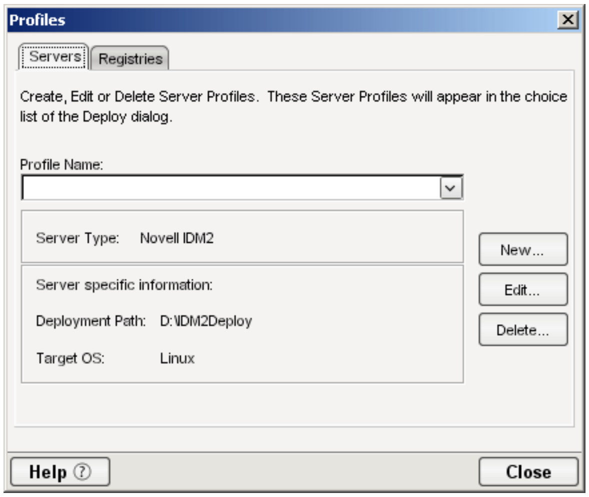
Name	Mo...
Execute	Jul
GetEvents	Jul
GetSchema	Jul
Initialize	Jul
InitializePublisher	Jul
InitializeSubscriber	Jul
Shutdown	Jul

The next step is to create a server profile for Identity Manager, which you will need when you deploy your project.

Creating a Server Profile

➢ **To create a server profile:**

1. Select **Profiles** from the Tools menu. The **Profiles** dialog box is displayed:



Create, Edit or Delete Server Profiles. These Server Profiles will appear in the choice list of the Deploy dialog.

Profile Name:

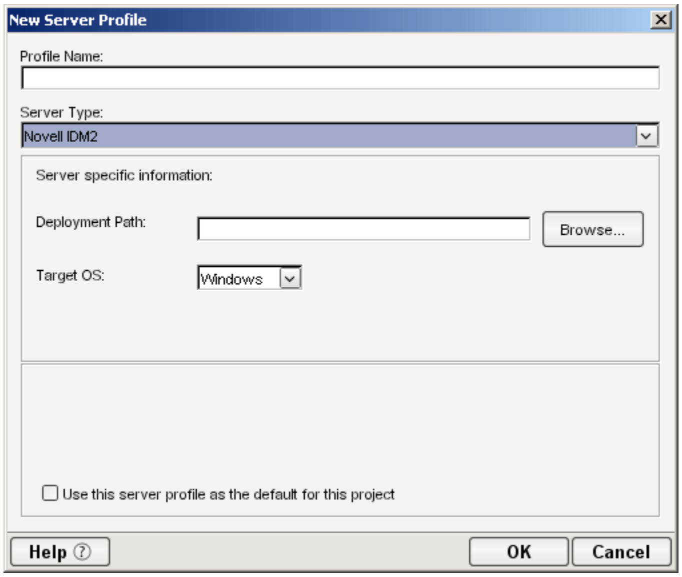
Server Type: Novell IDM2

Server specific information:

Deployment Path: D:\IDM2Deploy

Target OS: Linux

2. Select **New.** The **New Server Profile** dialog box is displayed:



New Server Profile

Profile Name:

Server Type: Novell IDM2

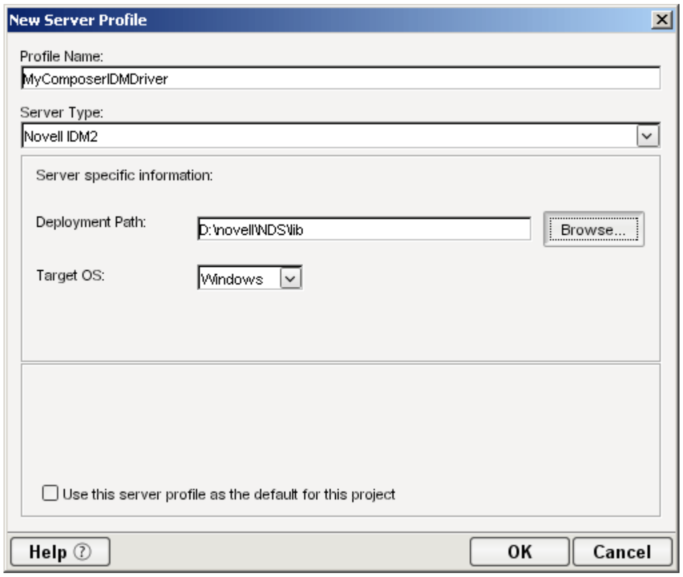
Server specific information:

Deployment Path:

Target OS: Windows

☐ Use this server profile as the default for this project

3. Type a name for the profile in the **Profile Name** field.
4. Select **Novell IDM2** from the **Server Type** list.
5. Use the browse button in the **Deployment Path** field to select the eDirectory "lib" directory (on Windows this is \NDS\lib; on Linux or UNIX this is /usr/lib/dirxml/classes).



New Server Profile

Profile Name: MyComposerIDMDriver

Server Type: Novell IDM2

Server specific information:

Deployment Path: D:\novell\NDS\lib

Target OS: Windows

☐ Use this server profile as the default for this project

6. Use the **Target OS** list to select the operating system to which you will deploy your driver.
7. Select **OK**. Composer returns you to the **Profiles** dialog box.
8. Select **Close**.

What is Included in the Integration Module for Enterprise - Custom Edition

The Integration Module for Enterprise - Custom Edition includes a set of xObjects that need to be part of any driver development project. These xObjects include:

- Web services - provided as entry points into the Composer action model
- XML Map Components
- XML Resources
- XML Templates
- Deployment xObject

The following sections describe these xObjects.

Web Services

The following table lists the Web service definitions supplied with the Integration Module for Enterprise - Custom Edition. These definitions are intended to serve as placeholders or stubs; you must customize them to work in your environment. The names of the Web services are very important (because they are used by the ComposerDriverShim.jar to call the services) and must not be changed.

Web Service	Description
Initialize	Sets up connection information used by the other services. Declares and sets session variables. Checks for loopback condition and creates a session variable to hold status of loopback condition check (true, false). **Note:** Although the DirXML engine prevents loopbacks, it's a best practice for efficient operation to check for and prevent loopbacks in the driver.
InitializePublisher	Initializes the Publisher channel. Checks session variable for the type of input (for example, “memory”, “file”) and performs appropriate setup tasks. Creates session variables for publisher options.
InitializeSubscriber	Initializes the Subscriber channel. Creates session variables for subscriber options.
Execute	Checks input for type of command (add, delete, modify, query), and performs the appropriate actions.
GetEvents	Used by the publisher to get events from a vector session variable.
GetSchema	Called by the ComposerDriverShim.jar. Returns the schema from the external application.
Shutdown	Removes variables created for the driver and stops services.

XML Map Components

XML Map components manipulate and transfer data from one XML document to another. The XML Maps are used by the Web services.

XML Map	Description
AddUser	Called by Execute. Uses an XML Interchange action to read data from an XML file into the DOM. Performs any necessary transformations, then uses another XML Interchange action to write the data to a file. In practice, the functionality in the component would probably be placed in an application-specific component (for example, a Siebel component). This component provides an example action model for adding a user to another system.
Check Stored events	Called by GetEvents. Checks for loopback condition.
Clean Stored events	Called by GetEvents. Used to clean up stored events to remove loopback condition.
DeleteUser	Called by Execute. Used to delete a user. Provides an example of using a Send Mail action to provide notification of the attempted deletion.
Map Connection Values	Called by Execute, Initialize, and Initialize Publisher. Used to map connection values to a Project Variable.
Modify	Called by Execute. Provides a placeholder for code to handle a Modify event. On the subscriber channel, this is a request to the external application to modify an entry. On the publisher channel, this is a notification to Identity Manager that an item in the external application has been modified.
Query	Called by Execute. Provides a placeholder for code to handle a Query event.
Store Events	Called by Execute. Used to store events.

XML Resources

The Integration Module for Enterprise - Custom Edition includes the following XML Resources. The XML Resources are provided as samples; they are not used in the project except where noted in the table.

XML Resource	Where Used
AddressSchema	
AddReply	
Address	
AddSample	
AddUser	
Association	Store Events
ClassUser	AddUser, GetSchema
Connection	
DeleteUser	
FullSchema	

XML Resource	Where Used
IdentifyReply	Execute
ndsModify3	
publish	GetEvents
Query	
reply	AddUser, DeleteUser, Modify, Query, Execute, Initialize, InitializePublisher, InitializeSubscriber, Shutdown
SampleDelete	
SampleInit	
SampleModify	
SampleOutputfromQuery	
SampleQuery	
UserSchema	GetSchema

XML Templates

XML templates contain sample documents, schemas, and stylesheets to assist you in designing and testing components. The following table lists the XML Templates included in the Integration Module for Enterprise - Custom Edition.

XML Template	Where Used
AddUser	AddUser (XML Map), Check Stored events, Clean Stored events, Store Events, Execute
Connection	Initialize, InitializePublisher, InitializeSubscriber
DeleteUser	DeleteUser (XML Map)
IdentQuery	
ndsModify	Modify
ndsModifyOut	
Query	
Reply	
Sampleschema	
Schema	GetSchema
UserSchema	

Deployment xObject

The Integration Module for Enterprise - Custom Edition includes a deployment xObject named “Demo”. The deployment xObject contains metadata about your deployment (information about which service triggers to create and which resources to deploy). You can view the deployment xObject by scrolling to the bottom of the Navigator frame and clicking on Deployment. The names of deployment xObjects that have been defined in the project are displayed below the Navigator frame. Right-click on “Demo” and select **Properties** to view the properties of the Demo deployment xObject.

What You Must Supply

- You must customize the artifacts supplied in the template (Web services, XML Maps, XML templates and resources, and deployment object) for your specific project.
- You must define Composer connection resources for the external application. For example, if you are integrating with a Siebel application, you must define a Siebel connection resource. Refer to the Composer Connect documentation for the external application with which you are integrating for information about creating a connection resource. For example, if you are integrating with a Siebel application, refer to the Novell *exteNd Composer 5.2.1 Connect for Siebel User's Guide*.
- You must author the components that provide the application-specific logic for the external application that you want to integrate with Identity Manager. Refer to the Composer Connect documentation for the external application with which you are integrating for information about creating an application-specific component. For example, if you are integrating with a Siebel application, refer to the Novell *exteNd Composer 5.2.1 Connect for Siebel User's Guide*.

3 Deploying the Driver

Deploying the Driver from Composer

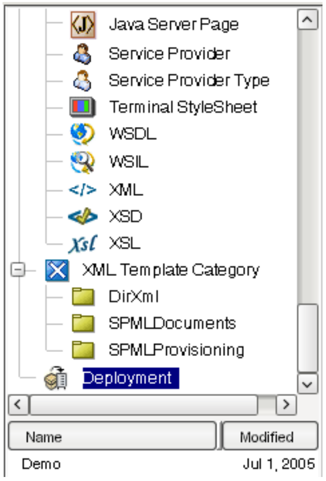
This section describes how to deploy the Integration Module for Enterprise - Custom Edition portion of your project. To deploy project components that connect to external applications, refer to the documentation for the Composer Connect for the external application, and to the *Novell exteNd Composer User's Guide*.

➢ To deploy the Integration Module for Enterprise - Custom Edition portion of your project:

Before you deploy the driver, make sure that you have set up a Server Profile and specified the deployment path (see "Creating a Server Profile" on page 17).

NOTE: For more information about deployment options, see "Deploying Your Project," in the *Novell exteNd Composer User's Guide*.

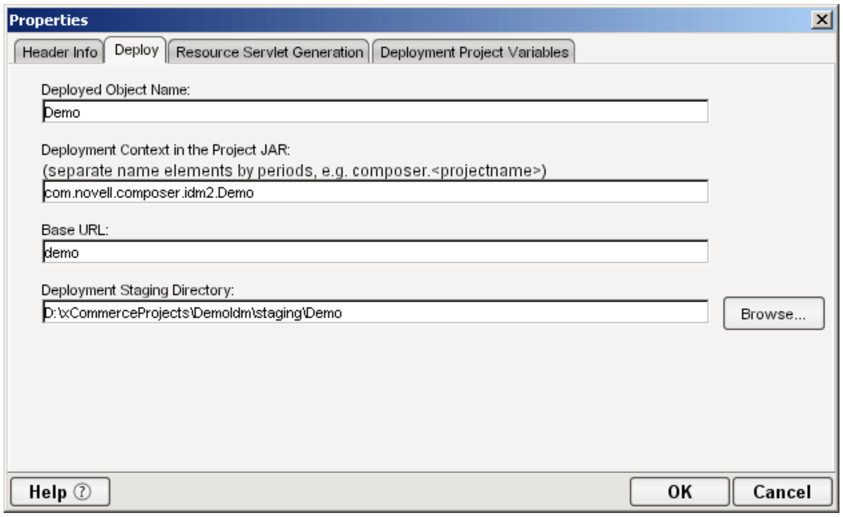
1 In the **Project** tab in the Navigator Frame, scroll down to the bottom and click on **Deployment**.



The Integration Module for Enterprise - Custom Edition includes a sample deployment xObject named "Demo". You will probably want to change this name to better describe your project.

2 Right-click on the Demo deployment xObject, and select **Properties**. Composer displays the deployment **Properties** dialog box.

3 Select the **Deploy** tab.



Deployed Object Name:
Demo

Deployment Context in the Project JAR:
(separate name elements by periods, e.g. composer.<projectname>)
com.novell.composer.idm2.Demo

Base URL:
demo

Deployment Staging Directory:
D:\xCommerceProjects\DemoIdm\staging\Demo

4 Composer fills in the **Deployment Object Name** field with the name of the deployment xObject. If desired, change the **Deployment Object Name** to a name that is appropriate for your project.

5 Composer fills in the **Deployment Context in the Project JAR** field with a default deployment context. If desired, change the deployment context in the **Deployment Context in the Project JAR** field to one that is appropriate for your application.

6 Composer fills in the **Base URL** field with a URL based on the deployment xObject name. If desired, change the **Base URL** to one that is appropriate for your project.

7 If desired, change the default path in the **Deployment Staging Directory** field. Composer uses this directory as a working directory during deployment, creating deployment objects and descriptor files.

8 Select **OK**.

9 In the **Navigator frame**, right-click on the name of the deployment object and select **Deploy Project**. The **Deployment** dialog box is displayed.

10 Select **Deploy.** Composer deploys the project using the deployment path specified in the server profile for the project.

NOTE: If you have previously deployed your driver, made a change, and redeployed, you may receive a message in the Composer System Output pane that some files (for example, xconfig.jar) could not be copied (“access denied”). This is because the files in question are already in the NDS lib directory from a previous deployment, and they are locked by eDirectory. If you have not made changes that affect the functionality of your Composer environment (for example, enabling a Connect; changing license strings for a Connect) you can ignore the message. If you have made changes that affect the functionality of your Composer environment, then your xconfig.jar has changed, and you will need to stop the eDirectory server, then repeat the steps in this procedure to deploy your project from Composer.

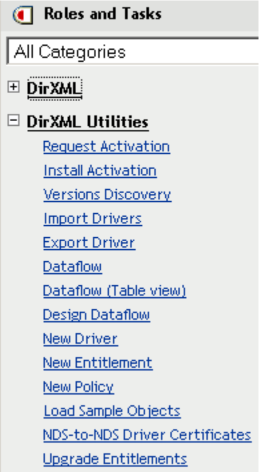
Configuring the Driver in Identity Manager

Creating the Driver Object

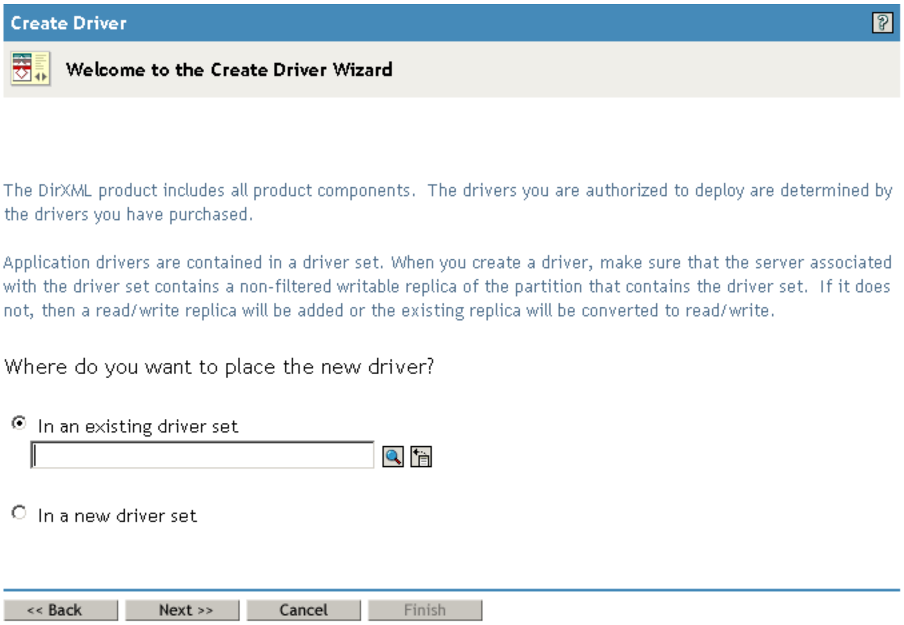
➢ **To create the driver object:**

1 Log into the instance of iManager that manages your eDirectory server.

2 Open the **DirXML Utilities** node in the iManager navigation frame.



3 Select **New Driver**. The Create Driver Wizard is displayed:



The DirXML product includes all product components. The drivers you are authorized to deploy are determined by the drivers you have purchased.

Application drivers are contained in a driver set. When you create a driver, make sure that the server associated with the driver set contains a non-filtered writable replica of the partition that contains the driver set. If it does not, then a read/write replica will be added or the existing replica will be converted to read/write.

Where do you want to place the new driver?

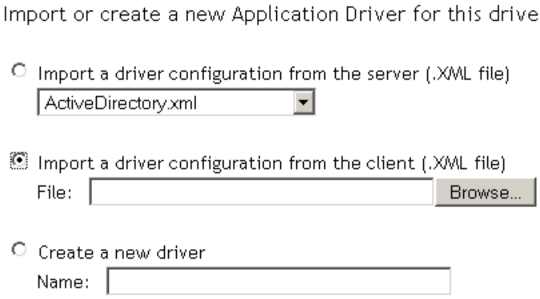
● In an existing driver set

○ In a new driver set

The next step is to select where you would like to create the new driver. You can create the driver in an existing driver set, or create a new driver set.

NOTE: When you set up a new driver, you can usually place the driver in an existing driver set. Use a new driver set for the first driver in a new Identity Manager installation. See the Identity Manager documentation for detailed information about driver sets and the iManager procedures for creating and configuring driver sets.

4 Select a driver set option, then select **Next.** The next screen in the **Create Driver Wizard** is displayed:



Import or create a new Application Driver for this driver set.

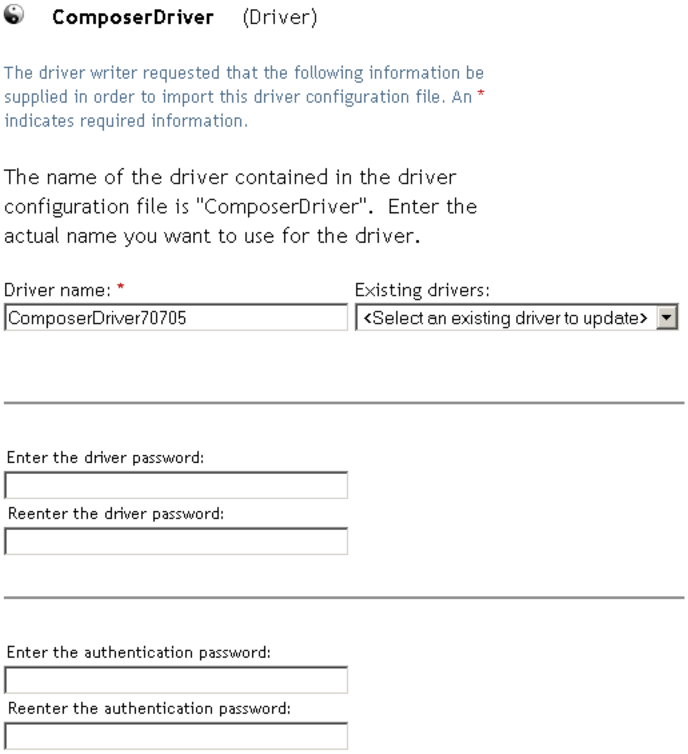
○ Import a driver configuration from the server (.XML file)
ActiveDirectory.xml

● Import a driver configuration from the client (.XML file)
File:

○ Create a new driver
Name:

5 Select the **Import a driver configuration from the client option**, then select the **Browse** button and navigate to the **ComposerDriver.xml** file in the \Novell\exteNd5\Composer\Designer\lib\IDM2 directory. Select the file, then select **Open** to return to the Create Driver Wizard.

6 Select **Next**. The Create Driver Wizard displays a page that you use to name the driver.



ComposerDriver (Driver)

The driver writer requested that the following information be supplied in order to import this driver configuration file. An * indicates required information.

The name of the driver contained in the driver configuration file is "ComposerDriver". Enter the actual name you want to use for the driver.

Driver name: * ComposerDriver70705

Existing drivers: <Select an existing driver to update>

Enter the driver password:

Reenter the driver password:

Enter the authentication password:

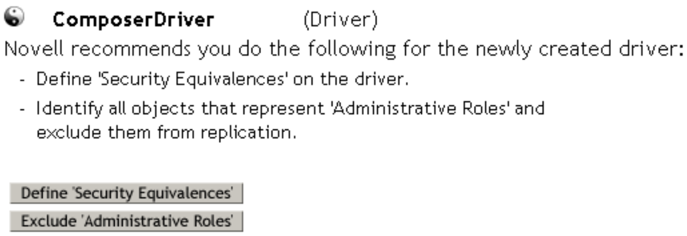
Reenter the authentication password:

The default name of the driver, from the configuration file, is ComposerDriver. While you can use the default name, you may want to choose a more meaningful name for your project.

7 If desired, type a new name for the driver in the **Driver name** field.

8 Leave the **Enter the driver password** and the **Enter the authentication password** fields blank.

9 Select **Next**. A message indicating that the driver configuration is being imported is displayed, then the next page of the **Create Driver** wizard is displayed:



ComposerDriver (Driver)

Novell recommends you do the following for the newly created driver:

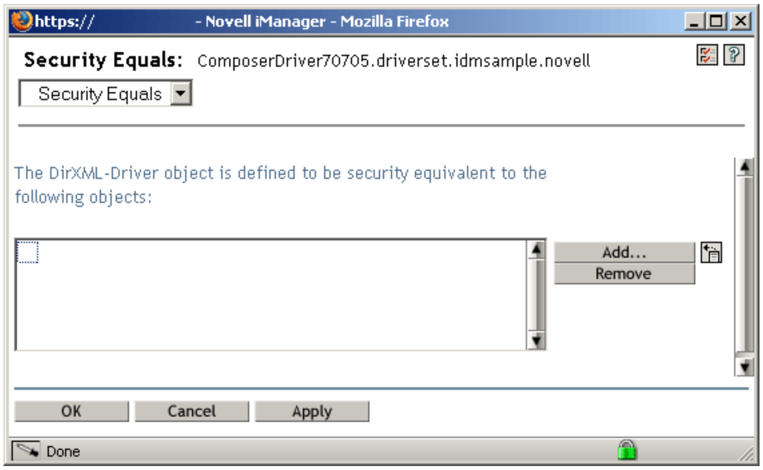
- Define 'Security Equivalences' on the driver.
- Identify all objects that represent 'Administrative Roles' and exclude them from replication.

Define 'Security Equivalences'
Exclude 'Administrative Roles'

The driver object must be granted sufficient eDirectory rights to any object that 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, post offices, resources, and distribution lists, and Create, Read, and Write rights to the post office container. Normally, the driver should be given security equal to Admin.

NOTE: The next two steps can be completed at a later time, if desired (by using the iManager driver administration features), but we strongly recommend that you complete these steps now.

10 Click **Define Security Equivalencies**. A new window is displayed:



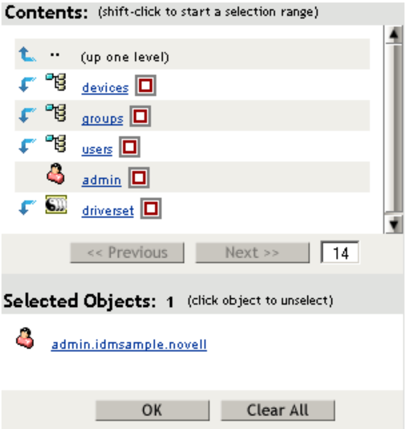
https:// - Novell iManager - Mozilla Firefox

Security Equals: ComposerDriver70705.driverset.idmsample.novell

Security Equals

The DirXML-Driver object is defined to be security equivalent to the following objects:

11 Select **Add**. A window is displayed that you use to select an object in the tree that has the appropriate level of rights that you would like to assign to this driver (for example, "admin").



Contents: (shift-click to start a selection range)

.. (up one level)
devices
groups
users
admin
driverset

Selected Objects: 1 (click object to unselect)

admin.idmsample.novell

12 Select an object from the tree, then select **OK**. You are returned to the previous window.

13 Select **OK**. You are returned to the Create Driver wizard.

14 Select **Exclude Administrative Roles**. The **Excluded Users** window is displayed.

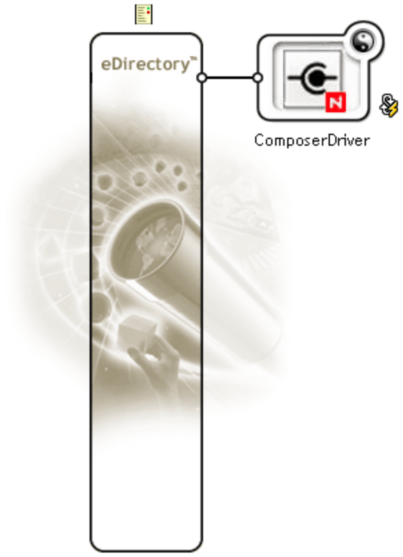
15 Select **Add**. A window is displayed that you use to browse the directory tree for users who should be excluded from having their data passed to the driver. Normally, you would exclude admin objects, since replicating their data across a driver connection is not good practice in most cases.

16 Select the administrative roles that you want to exclude, then select **OK**. You are returned to the previous window.

17 Select **OK**. You are returned to the **Create Driver** wizard.

18 Select **Next.** A driver summary page is displayed.

19 Select finish with Overview. A graphical representation of the driver in eDirectory is displayed:



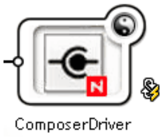
NOTE: You can view this screen again at any time by using the **DirXML Overview** link under DirXML Management in the iManager navigation tree.

The new driver appears as a large icon connected to the eDirectory trunk. The next step is to configure driver settings.

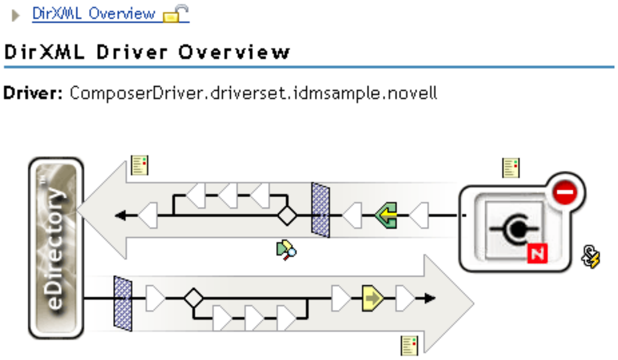
Configuring Driver Settings

➤ **To configure driver settings:**

1. If the DirXML overview screen is not currently displayed, display it by selecting the **DirXML Overview** link under DirXML Management in the iManager navigation tree.
2. Click the large driver icon:

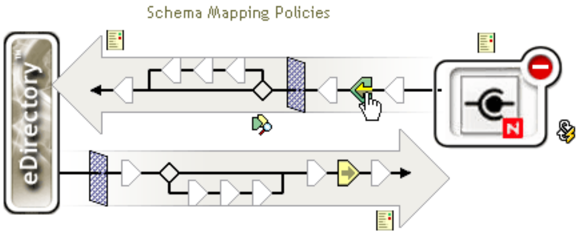
   The **DirXML Driver Overview** is displayed:

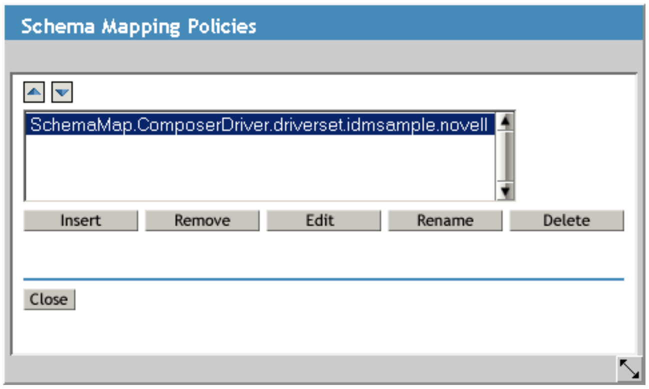
   DirXML Driver Overview

   Driver: ComposerDriver.driverset.idmsample.novell

   The top horizontal arrow represents the Publisher channel and the bottom horizontal arrow represents the Subscriber channel. As you pass the mouse pointer over an object in the graphic, a description of the object is displayed:

3. Click on the **Schema Mapping Policies** icon for the Publisher channel. The **Schema Mapping Policies** dialog box is displayed:

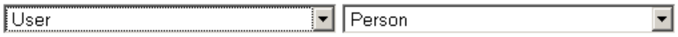
   

   SchemaMap.ComposerDriver.driverset.idmsample.novell

4. Select **Edit**. The **DirXML Policy** dialog box is displayed.

5 Select **Refresh Application Schema**.

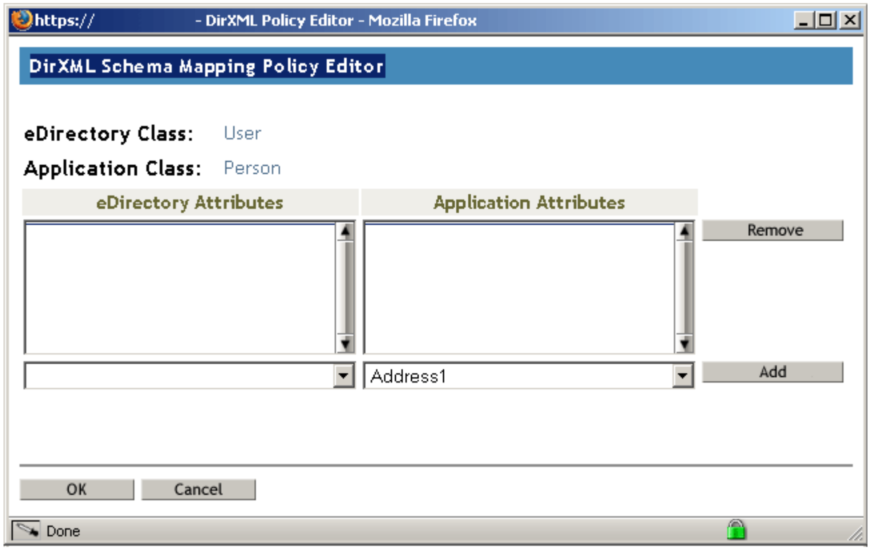
6 Select **Add**, then select the classes to map from the drop down lists under the **eDirectory Classes** and **Application Classes** lists:



User

Person

7 Select **Attributes.** The **DirXML Schema Mapping Policy Editor** is displayed:



https:// - DirXML Policy Editor - Mozilla Firefox

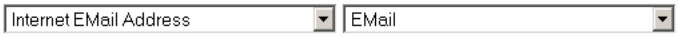
DirXML Schema Mapping Policy Editor

eDirectory Class: User

Application Class: Person

eDirectory Attributes	Application Attributes
	Address1

8 Select an attribute from the drop down list in the **Application Attributes** column; then select an attribute to which to map the application attribute from the drop down list in the **eDirectory Attributes** column.

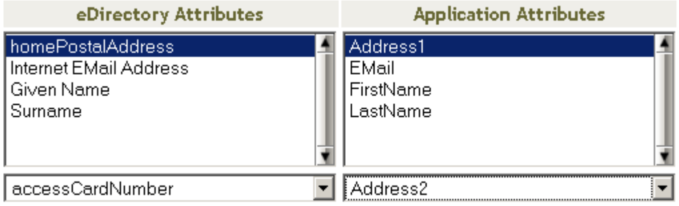


Internet EMail Address

EMail

9 Select **Add.**

10 Repeat Step 8 and Step 9 until you have mapped all of the attributes that you need to map.



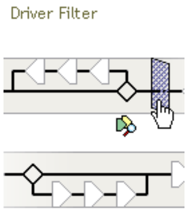
eDirectory Attributes	Application Attributes
homePostalAddress	Address1
Internet EMail Address	EMail
Given Name	FirstName
Surname	LastName
accessCardNumber	Address2

11 Select **OK.** You are returned to the **DirXML Policy** dialog box.

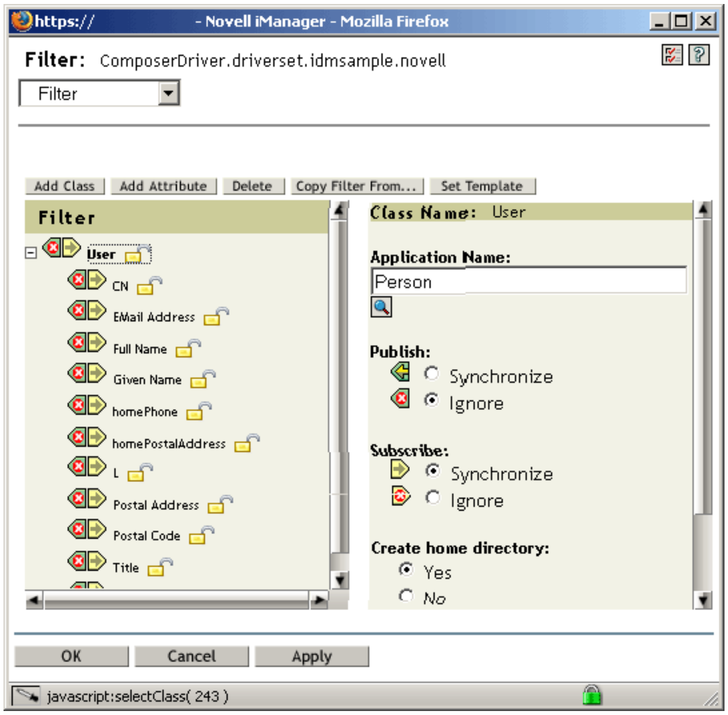
12 Select **OK**. You are returned to the **Schema Mapping Policies** dialog box.

13 Select **Close**. You are returned to the **DirXML Driver Overview** page.

14 Click the **Driver Filter** icon for the Publisher channel.



The filter window is displayed:



Filter: ComposerDriver.driverset.idmsample.novell

Filter

Filter
- User
  - CN
  - EMail Address
  - Full Name
  - Given Name
  - homePhone
  - homePostalAddress
  - L
  - Postal Address
  - Postal Code
  - Title

Class Name: User

Application Name: Person

Publish:
○ Synchronize
● Ignore

Subscribe:
● Synchronize
○ Ignore

Create home directory:
● Yes
○ No

javascript:selectClass(243)

Filters enable you to specify the classes and attributes that an application can send to and receive from eDirectory. The read-only **Filter** list on the left shows the attributes of the class. The **Class Name** list on the right displays options associated with the target object. Click the help button in the filter window for more information about the features of this window.

15 Specify the attributes that the application can send to and receive from eDirectory. Select **Synchronize** for an attribute if you want changes to attribute values to be reported and synchronized. Select **Ignore** if you do not want changes to attribute values to be reported and synchronized.

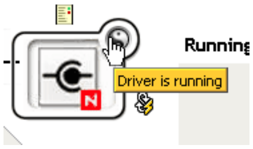
16 Select **OK.** You are returned to the **DirXML Driver Overview** page.

17 Repeat Step 3 through Step 16 for the Subscriber channel.

18 Click on the driver icon (on the right side of the page) to display the **Modify Object** dialog box.

19 Check that the driver parameters are correct for your environment.

20 To start or stop the driver, click on the rounded “status indicator” in the upper right corner of the driver icon, and select the appropriate command from the menu.



Index