SUN SEEBEYOND

eWAY™ LDAP ADAPTER USER'S GUIDE

Release 5.1.3

Part Number: 820-0993

Version 20070420092716

Contents

Chapter 5

Chapter 6

Chapter 1

Introducing the LDAP eWay

This guide explains how to use, set properties for, and operate the Sun SeeBeyond eWay™ LDAP Adapter, referred to as the LDAP eWay throughout this guide.

This chapter provides a brief overview of operations, components, general features, and system requirements of the eWay.

What's in This Chapter

1.1 About LDAP

LDAP (Lightweight Directory Access Protocol), is an Internet protocol for accessing information directories. A directory service is a distributed database application designed to manage the entries and attributes in a directory. LDAP runs over TCP/IP.

LDAP allows clients to access different directory services based on entries. It makes the entries, along with their attributes and values, available to users and other applications, on a controlled-access basis.

The LDAP OTD provides access to the operations available via the LDAP protocol. To give you a better understanding of these operations and how they are implemented in the OTD, this section briefly summarizes how LDAP works.

1.1.1 Entries, Attributes, and Values

An LDAP directory has entries that contain information pertaining to some entity. Each of the entry's attributes has a name and one or more values. The names of attributes are most often mnemonic strings, such as **cn** for common name, or **mail** for e-mail address.

For example, a company may have an employee directory. Each entry in the employee directory represents an employee. The employee entry contains such information as the name, e-mail address, and phone number, as shown in the following example:

```
cn: John Doe
mail: johndoe@sun.com
mail: jdoe@stc.com
telephoneNumber: 471-6000 x.1234
```

Each part of the descriptive information, such as an employee's name, is known as an attribute. In the example above, the Common Name (**cn**) attribute, represents the name of the employee. The other attributes are **mail** and **telephoneNumber**.

Each attribute can have one or more values. For example, an employee entry may contain a mail attribute whose values are **johndoe@sun.com** and **jdoe@stc.com**. In the previous example, the mail attribute contains two mail values.

1.1.2 LDAP Directory Structure

The organization of a directory is a tree structure. The topmost entry in a directory is known as the root entry. This entry normally represents the organization that owns the directory.

Entries at the higher level of hierarchy, represent larger groupings or organizations. Entries under the larger organizations represent smaller organizations that make up the larger ones. The leaf nodes (or entries) of the tree structure represent the individual persons or resources.

1.1.3 Distinguished Names and Relative Distinguished Names

An entry is made up of a collection of attributes that have a unique identifier called a distinguished name (DN). A DN consists of a name that uniquely identifies the entry at that hierarchical level. In the example above, John Doe and Jane Doe are different common names (**cn**) that identify different entries at that same level.

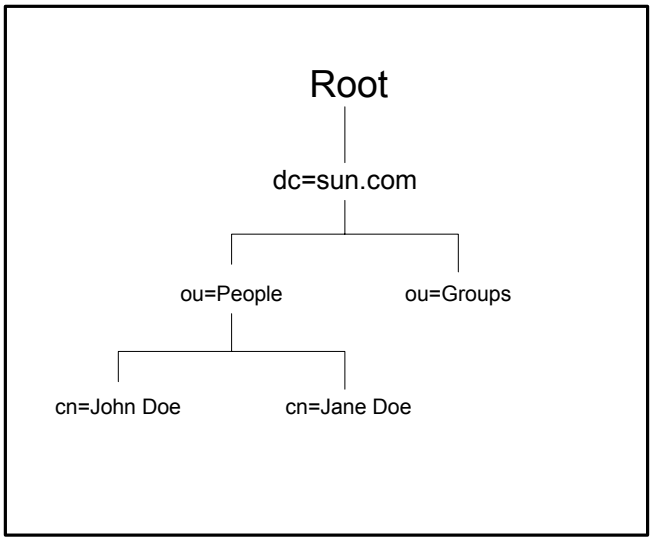
A DN is also a fully qualified path of names that trace the entry back to the root of the tree. For example, the distinguished name of the John Doe entry is:

```
cn=John Doe, ou=People, dc=sun.com
```

A relative distinguished name (RDN) is a component of the distinguished name. For example, **cn=John Doe, ou=People** is a RDN relative to the root RDN **dc=sun.com**. DNs are used to describe the fully qualified path to an entry while an RDN is used to describe the partial path to the entry relative to another entry in the tree.

Figure 1 illustrates an example of an LDAP directory structure with distinguished names and relative distinguished names.

Figure 1 LDAP Directory Structure



Wherever necessary, the LDAP OTD mimics this same directory structure.

1.1.4 LDAP Service and LDAP Client

A directory service is a distributed database application designed to manage the entries and attributes in a directory. A directory service also makes the entries and attributes available to users and other applications. OpenLDAP server is an example of a directory service. Other directory services include Sun Active Directory Service (Sun Microsystems) and Microsoft Active Directory.

A directory client accesses a directory service using the LDAP protocol. A directory client may use one of several client APIs available in order to access the directory service.

1.1.5 Referrals

The native APIs developed for the LDAP eWay query the results of a search based on specified criteria. The search results may consist of a number of referrals.

A referral is an entity that is used to redirect a client's request to another server. A referral contains the names and locations of other objects. For example, an LDAP server sends a referral to the client to indicate that the information that the client has requested can be found at another location (or locations), possibly at another server or several servers.

The referral contains the URL of the LDAP server that holds the actual entry. The LDAP URL contains the server's host/port and an object's DN. For instructions on how to set the eWay properties for referrals, see **Configuring the Referrals Section Properties** on page 42.

1.2 About the LDAP eWay

This section describes the general information about the LDAP eWay and its operation with Sun SeeBeyond eGate™ Integrator, also referred throughout this book as eGate Integrator.

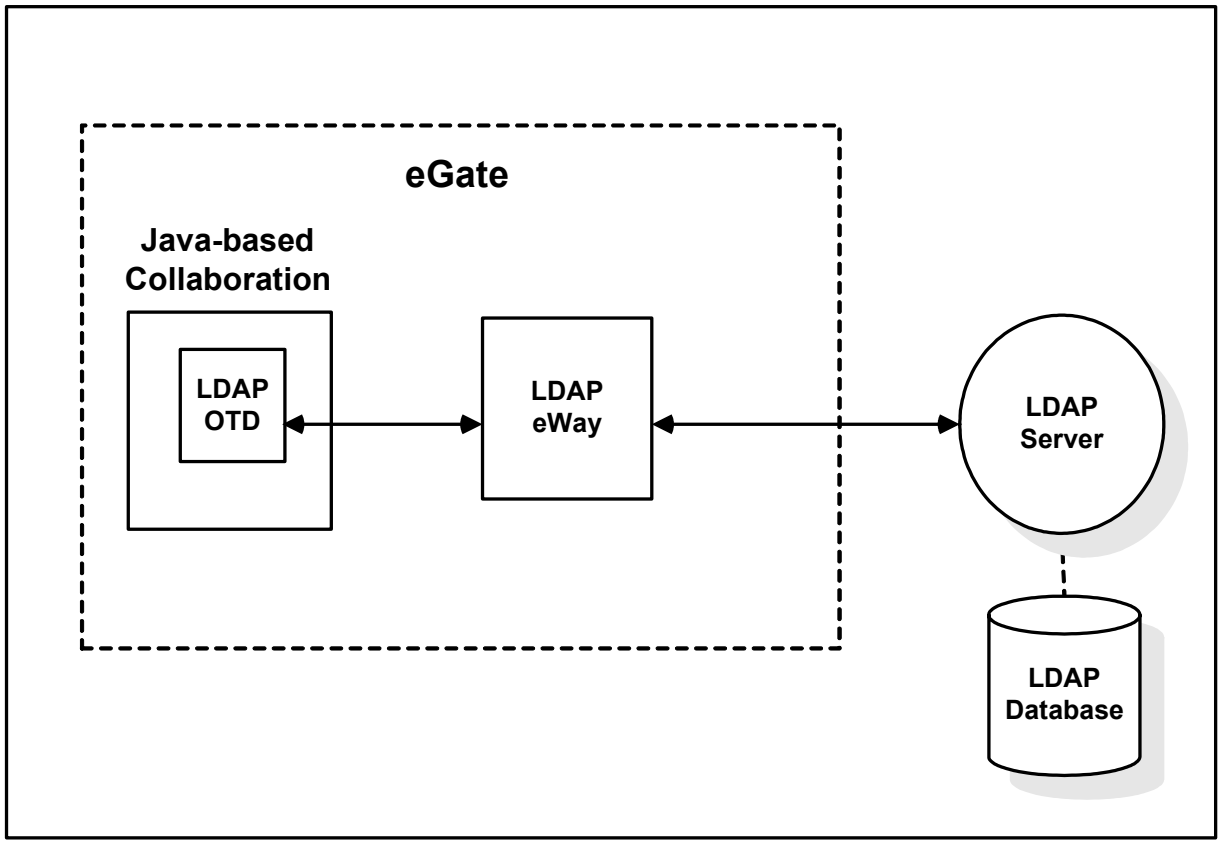
1.2.1 eWay General Operation

The LDAP eWay enables eGate to exchange data with an LDAP directory on an LDAP server. The eWay consists of two components, an LDAP connector and an LDAP Object Type Definition (OTD). The OTD utilizes the connector to connect to a particular LDAP server.

By connecting to an LDAP server, the eWay enables eGate to search, compare, and modify an LDAP directory using the LDAP protocol. The eWay utilizes the LDAP OTD to perform these functions. This OTD carries LDAP information through eGate and allows the information to be processed by eGate's Java-based Collaborations.

See Figure 2 for a general diagram of the architecture of the LDAP eWay.

Figure 2 LDAP eWay Architecture



In addition, the LDAP OTD exposes the application programming interface (API) for accessing the LDAP directory. The LDAP OTD enables you to create Java-based

Collaboration Definitions that execute LDAP operations, for example, searching an LDAP directory, adding entries to the directory, and modifying entries in the directory.

A given instance of an LDAP OTD uses only one instance of an LDAP connector. You can use as many instances of the LDAP OTD in a single data-exchange scenario, as necessary.

1.2.2 Java Naming and Directory Interface

The LDAP eWay uses Sun Microsystem's Java Naming and Directory Interface (JNDI) LDAP provider. This set of APIs allows a Java program to store objects and look up objects using multiple naming services in a standard manner.

The JNDI is included in the Java 2 Software Developer's Kit (SDK) version 1.4 installed as part of eGate.

1.2.3 Third-Party License File Agreement

A disclaimer readme file is available for review when you install the LDAP eWay. The disclaimer is applicable to the jCookie Library, a robust and easy to use library for client-side HTTP state management in Java applications.

After successful installation, you can view the following third-party file using any text file viewer:

LDAPeWay_THIRDPARTYLICENSEREADME.txt

Third-party license files are located at:

```
\repository\ThirdPartyLicenses
```

where repository indicates the folder where the eGate Repository is installed.

1.3 What's New in This Release

The eWay LDAP Adapter includes the following new features:

What's New in Version 5.1.3

- The Secure Sockets Layer (SSL) feature is available, which enables LDAP data exchanges to be secure from unauthorized interception from "hackers" or other entities.

What's New in Version 5.1.2

- In version 5.1.2 the Connection and Security sections have been copied to the LDAP External System Environment properties window.
- Certain ERROR messages are now logged as DEBUG messages in server log file. These errors are user data related, for example:

  [LDAP: error code 32 - No Such Object]

You can review these messages by setting the Module Log Level to FINE or higher for the LDAP eWay in the Integration Server Administration console. You can catch and handle LDAPApplicationException messages in your code.

- "ReturningObjFlag" is added as part of SearchOptions. Please refer to the Javadoc for details.

What's New in Version 5.1.1

There are no new features in this release.

What's New in Version 5.1

- The LDAP eWay no longer requires manually importing the **stcldap14.jar** file to properly catch the following exception message:

  ```
  com.stc.connector.appconn.ldap.LDAPApplicationException
  ```

- Version Control: An enhanced version control system allows you to effectively manage changes to the eWay components.
- Multiple Drag-and-Drop Component Mapping from the Deployment Editor: The Deployment Editor now allows you to select multiple components from the Editor's component pane, and drop them into your Environment component.
- Support for Runtime LDAP Configuration: eWay configuration properties now support LDAP key values.
- MDB Pool Size Support: Provides greater flow control (throttling) by specifying the maximum and minimum MDB pool size.
- Connection Retry Support: Allows you to specify the number of attempts to reconnect, and the interval between retry attempts, in the event of a connection failure.
- Connectivity Map Generator: Generates and links your Project's Connectivity Map components using a Collaboration or Business Process.

1.4 What's In this Document

This document includes the following chapters:

- **Chapter 1 "Introducing the LDAP eWay"**: Provides an overview description of the LDAP eWay, as well as high-level information about this document.
- **Chapter 2 "Installing the LDAP eWay"**: Describes the system requirements and provides instructions for installing the LDAP eWay.
- **Chapter 4 "Setting LDAP eWay Properties"**: Provides instructions for configuring the eWay to communicate with **LDAP**.
- **Chapter 5 "Using the LDAP OTD"**: Provides instructions for creating Object Type Definitions to be used with the LDAP eWay.
- **Chapter 6 "Reviewing the Sample Project"**: Provides instructions on using LDAP eWay operations in the Java Collaboration Definition (JCD).

1.4.1 Scope

This document describes the process of installing, configuring, and running the LDAP eWay.

This document does not cover the Java methods exposed by this eWay. For information on the Java methods, download and view the LDAP eWay Javadoc files from the Enterprise Manager.

1.4.2 Intended Audience

This guide is intended for experienced computer users who have the responsibility of helping to set up and maintain a fully functioning Java Composite Application Platform Suite system. This person must also understand any operating systems on which the Java Composite Application Platform Suite will be installed (Windows and UNIX), and must be thoroughly familiar with Windows-style GUI operations.

1.4.3 Text Conventions

The following conventions are observed throughout this document.

Table 1 Text Conventions

Text Convention	Used For	Examples
Bold	Names of buttons, files, icons, parameters, variables, methods, menus, and objects	▪ Click **OK**. ▪ On the **File** menu, click **Exit**. ▪ Select the **eGate.sar** file.
`Monospaced`	Command line arguments, code samples; variables are shown in ***bold italic***	`java -jar ***filename***.jar`
Blue bold	Hypertext links within document	See **Text Conventions** on page 13
Blue underlined	Hypertext links for Web addresses (URLs) or email addresses	http://www.sun.com

1.4.4 Related Documents

The following Sun documents provide additional information about the Java Composite Application Platform Suite product:

- *Sun SeeBeyond eGate™ Integrator User's Guide*
- *Composite Application Platform Suite Installation Guide*

1.5 Sun Microsystems, Inc. Web Site

The Sun Microsystems web site is your best source for up-to-the-minute product news and technical support information. The site's URL is:

http://www.sun.com

1.6 Documentation Feedback

We appreciate your feedback. Please send any comments or suggestions regarding this document to:

CAPS_docsfeedback@sun.com

Chapter 2

Installing the LDAP eWay

This chapter explains how to install the LDAP eWay, access the accompanying documentation and sample Projects.

What's in This Chapter

2.1 Installing the LDAP eWay

The Java Composite Application Platform Suite Installer, referred to throughout this guide as the Suite Installer, is a web-based application that is used to select and upload core products, composite applications, and add-on files (eWays). The following section describes how to install the components required for this eWay.

Refer to the readme for the latest information on:

- Supported Operating Systems
- System Requirements
- External System Requirements

Note: *When the Repository is running on a UNIX operating system, the eWays are loaded from the Enterprise Manager running on a Windows platform connected to the Repository server using Internet Explorer.*

2.1.1 Installing the LDAP eWay on a Java CAPS system

Follow the directions for installing the Java Composite Application Platform Suite (Java CAPS) in the *Composite Application Platform Suite Installation Guide.*

After you have installed eGate, do the following:

1. From the Suite Installer, click the Administration tab, and then click the link to install additional products.

2 Select the products for your Java Composite Application Platform Suite, and include the following:

- **FileeWay** (the File eWay is used by most sample Projects)
- **LDAPeWay**

To upload the LDAP eWay User's Guide, Help file, Javadoc, Readme, and sample Projects, select the following:

- **LDAPeWayDocs**

3 Once you have selected all of your products, click **Next** in the top-right or bottom-right corner of the **Select Java Composite Application Platform Suite Products to Install** box.

4 From the **Selecting Files to Install** box, locate and select your first product's SAR file. Once you have selected the SAR file, click **Next**. Your next selected product appears. Follow this procedure for each of your selected products. The **Installation Status** window appears and installation begins after the last SAR file has been selected.

5 Once your product's installation is finished, continue installing the Java Composite Application Platform Suite as instructed in the *Composite Application Platform Suite Installation Guide.*

Adding the eWay to an Existing Java Composite Application Platform Suite Installation

It is possible to add the eWay to an existing Java Composite Application Platform Suite installation.

Steps required to add an eWay to an Existing Java CAPS installation include:

1 Complete steps 1 through 4 above. in **Installing the LDAP eWay on a Java CAPS system** on page 15

2 Once your product's installation is finished, open the Enterprise Designer and select **Update Center** from the Tools menu. The **Update Center Wizard** appears.

3 For Step 1 of the wizard, simply click **Next**.

4 For Step 2 of the wizard, click the **Add All** button to move all installable files to the **Include in Install** field, then click **Next**.

5 For Step 3 of the wizard, wait for the modules to download, then click **Next**.

6 The wizard's Step 4 window displays the installed modules. Review the installed modules and click **Finish.**

7 When prompted, restart the IDE (Integrated Development Environment) to complete the installation.

After Installation

Once you install the eWay, it must then be incorporated into a Project before it can perform its intended functions. See the *Sun SeeBeyond eGate™ Integrator User's Guide* for more information on incorporating the eWay into an eGate Project.

2.1.2 Extracting the Sample Projects and Javadocs

The LDAP eWay includes sample Projects and Javadocs. The sample Projects are designed to provide you with a basic understanding of how certain LDAP operations are performed using the eWay, while Javadocs provide a list of classes and methods exposed in the eWay.

Steps to extract the Javadoc include:

1. Click the Documentation tab of the Suite Installer, then click the Add-ons tab.
2. Click the Sun SeeBeyond eWay LDAP Adapter link. Documentation for the LDAP eWay appears in the right pane.
3. Click the icon next to **Javadoc** and extract the ZIP file.
4. Open the index.html file to view the Javadoc.

Steps to extract the Sample Projects include:

1. Click the Documentation tab of the Suite Installer, then click the Add-ons tab.
2. Click the Sun SeeBeyond eWay LDAP Adapter link. Documentation for the LDAP eWay appears in the right pane.
3. Click the icon next to **Sample Projects** and extract the ZIP file.

   Refer to **Importing a Sample Project** on page 84 for instructions on importing the sample Project into your repository via the Enterprise Designer.

2.2 ICAN 5.0 Project Migration Procedures

This section describes how to transfer your current ICAN 5.0 Projects to Sun Java Composite Application Platform Suite, version 5.1.3. Only Projects developed on ICAN version 5.0.2 and above can be migrated successfully to the Sun Java Composite Application Platform Suite. To migrate your ICAN 5.0 Projects, do the following:

Export the Project

1. Before you export your Projects, save your current ICAN 5.0 Projects to your Repository.
2. From the Project Explorer, right-click your Project and select **Export** from the shortcut menu. The Export Manager appears.
3. Select the Project that you want to export in the left pane of the Export Manager and move it to the Selected Projects field by clicking the **Add to Select Items** (arrow) button, or click **All** to include all of your Projects.

4 In the same manner, select the Environment that you want to export in the left pane of the Export Manager and move it to the Selected Environments field by clicking the **Add to Select Items** (arrow) button, or click **All** to include all of your Environments.

5 Browse to select a destination for your Project ZIP file and enter a name for your Project in the **ZIP file** field.

6 Click **Export** to create the Project ZIP file in the selected destination.

Install Sun Java Composite Application Platform Suite

7 Install the Sun Java Composite Application Platform Suite, including all eWays, libraries, and other components used by your ICAN 5.0 Projects.

8 Start the Sun SeeBeyond Enterprise Designer.

Import the Project

9 From the Enterprise Designer's Project Explorer tree, right-click the Repository and select **Import Project** from the shortcut menu. The Import Manager appears.

10 Browse to and select your exported Project file.

11 Click **Import**. A warning message, **"Missing APIs from Target Repository**," may appear at this time. This occurs because various product APIs were installed on the ICAN 5.0 Repository when the Project was created, that are not installed on the Sun Java Composite Application Platform Suite Repository. These APIs may or may not apply to your Projects. You can ignore this message if you have already installed all of the components that correspond to your Projects. Click **Continue** to resume the Project import.

12 Close the Import Manager after the Project is successfully imported.

Deploy the Project

13 A new Deployment Profile must be created for each of your imported Projects. When a Project is exported, the Project's components are automatically *"checked in"* to Version Control to write-protected each component. These protected components appear in the Explorer tree with a red padlock in the bottom-left corner of each icon. Before you can deploy the imported Project, the Project's components must first be *"checked out"* of Version Control from both the Project Explorer and the Environment Explorer. To *"check out"* all of the Project's components, do the following:

A From the Project Explorer, right-click the Project and select **Version Control > Check Out** from the shortcut menu. The Version Control - Check Out dialog box appears.

B Select **Recurse Project** to specify all components, and click **OK**.

C Select the Environment Explorer tab, and from the Environment Explorer, right-click the Project's Environment and select **Version Control > Check Out** from the shortcut menu.

D Select **Recurse Environment** to specify all components, and click **OK**.

14 If your imported Project includes File eWays, these must be reconfigured in your Environment prior to deploying the Project. To reconfigure File eWays, do the following:

A The Environment File External System properties can now accommodate both inbound and outbound eWays. If your previous Environment includes both inbound and outbound File External Systems, delete one of these (for example, the outbound File External System).

B From the Environment Explorer tree, right-click your remaining File External System, and select **Properties** from the shortcut menu. The Properties Editor appears.

C The Directory property has been relocated from the Connectivity Map Properties to the Environment Properties. Set the inbound and outbound Directory values, and click **OK**.

15 If your imported Project includes LDAP eWays, these must be reconfigured in your Environment prior to deploying the Project. To reconfigure LDAP eWays, do the following:

A From the Environment Explorer tree, right-click your remaining LDAP External System, and select **Properties** from the shortcut menu. The Properties Editor appears.

B Please refer to **Configuring the eWay Environment Properties** on page 39 for configuration details. Click **OK** to finish.

2.3 Java CAPS 5.1.0 and 5.1.1 Upgrade Procedures

5.1.3 Sun SeeBeyond LDAP eWay Adapter changes include copying Connection and Security sections from the Connectivity Map properties window to the LDAP External System Environment properties window. This change improves backwards compatibility support for eGate, eXchange, or other Sun SeeBeyond products which use the LDAP eWay.

2.3.1 5.0.x to 5.1.3 Upgrade Procedures

There are no special requirements when importing 5.0.x projects into 5.1.3. Standard ICAN 5.0 project migration procedures apply under these conditions. Refer to **ICAN 5.0 Project Migration Procedures** on page 17 for more details.

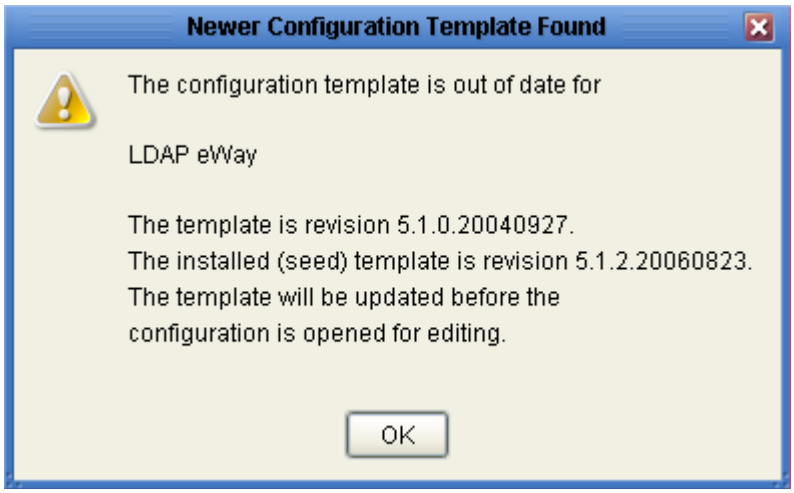
2.3.2 5.1.0 or 5.1.1 to 5.1.3 Upgrade Procedures

There are new versions of the Configuration templates used in 5.1.3. For previous 5.1.0 or 5.1.1 projects that are imported or going through an "in-place upgrade" to the latest version, the Configuration template will be upgraded during design time or build time.

At design time when you open the Connectivity Map or Environment properties window, a warning window appears, see **Figure 3 on page 20**, and the Configuration

template automatically upgrades. The Connection and Security sections in the Connectivity Map retain the original values from previous 5.1.x version. Both sections in the Environment properties window are populated with default values. You can now update the Environment properties with any necessary change, and run the project.

Figure 3 Configuration Template Warning Window



If you attempt to build a project without first opening either the Connectivity Map or Environment property window, code generation will automatically upgrade the Configuration template. Once this build-time upgrade scenario is complete, you will not see the warning window anymore, as shown in Figure 3.

2.3.3 Configuration Precedence

There are two sets of Connection and Security configuration sections in the Connectivity Map and Environment properties windows. Values set in the following three Environment properties act as a trigger during build time.

These parameters are as follows:

- Environment Configuration -> Connection -> ProviderURL
- Environment Configuration -> Connection -> Principal
- Environment Configuration -> Connection -> Credentials

If any of above three parameters holds a value in the Environment properties window, both Connection and Security configuration in the Environment will be used for code generation, and the Connection and Security sections in the Connectivity Map are ignored. On the other hand, if all above parameters are empty, then code generation will ignore the Environment sections and use both configuration sections from the Connectivity Map for connection.

2.4 Installing Enterprise Manager eWay Plug-Ins

The **Sun SeeBeyond Enterprise Manager** is a Web-based interface that allows you to monitor and manage your Java Composite Application Platform Suite applications. The Enterprise Manager requires an eWay specific "plug-in" for each different eWay you install. These plug-ins enable the Enterprise Manager to target specific alert codes for each eWay type.

The *Composite Application Platform Suite Installation Guide* describes how to install Enterprise Manager. The *Sun SeeBeyond eGate Integrator System Administration Guide* describes how to monitor servers, Services, logs, and alerts using the Enterprise Manager and the command-line client.

The **eWay Enterprise Manager plug-ins** are available from the **List of Components to Download** under the Suite Installer's **Downloads** tab. The plug-in required for LDAP is listed as the **LDAP eWay Enterprise Manager Plug-in**.

The following steps are required to install eWay plug-ins into the Enterprise Manager:

1. From the **Enterprise Manager's** Explorer toolbar, click the **Configuration** icon.
2. Click the **Web Applications Manager** tab, go to the **Auto-Install from Repository** sub-tab, and connect to your Repository.
3. Select the application plug-ins you require, and click **Install**. The application plug-ins are installed and deployed.

Alternately, you can install eWay plug-ins using the following steps:

1. From the Suite Installer's **Downloads** tab, select the Plug-Ins you require and save them to a temporary directory.
2. From the **Enterprise Manager's** Explorer toolbar, click the **Configuration** icon.
3. Click the **Web Applications Manager** tab and go to the **Manage Applications** sub-tab.
4. Browse for and select the WAR file for the application plug-in that you downloaded, and click **Deploy**. The plug-in is installed and deployed.

Viewing Alert Codes

You can view and delete alerts using the Enterprise Manager. An Alert is triggered when a specified condition occurs in a Project component. The purpose of the Alert is to warn the administrator or user that a condition has occurred.

To View the eWay Alert Codes

1. Add the eWay Enterprise Manager plug-in for this eWay.
2. From the **Enterprise Manager**'s Explorer toolbar, click the **Configuration** icon.
3. Click the **Web Applications Manager** tab and go to the **Manage Alert Codes** sub-tab. Your installed eWay alert codes display under the **Results** section.

For information on Managing and Monitoring alert codes and logs, as well as how to view the alert generated by the project component during runtime, see the *Sun SeeBeyond eGate™ Integrator System Administration Guide.*

Table 2 Alert Codes for the LDAP eWay

Alert Code\Description	Description Details	User Actions
LDAP-CONNECTIONFAILED=Failed to establish connection to LDAP server on server.	Unable to establish a connection to the LDAP server. You have reached the maximum connection retry limit.	▪ LDAP server is down; start your server. ▪ External configuration information is invalid. You may need to verify the following : ♦ Authentication ♦ Credentials ♦ Principal ♦ Provider URL
LDAP-DISCONNECTIONFAILED=Failed to disconnect from LDAP server.	Unable to close the external system connector and release resources.	▪ LDAP server is down; start your server.
LDAP-CLEANUPFAILED=Failed to clean up LDAP eWay connection handler.	Failed to clean up any resources or reset any state held by the LDAP eWay Connection instance.	▪ Contact Support.
LDAP-INITIALIZEFAILED=Failed to initialize LDAP eWay connection.	Unable to initialize a connection to the LDAP server.	▪ This is a general exception when a connection to the LDAP server has failed to initialize. You may need to verify the following : ♦ Connection Retry ♦ Connection Retry Interval ♦ Number of Retries ♦ External configuration information (Authentication, Credentials, Principal, and Provider URL)

Note: *An alert code is a warning that an error has occurred. It is not a diagnostic. The user actions noted above are just some possible corrective measures you may take. Refer to the log files for more information. For information on Managing and Monitoring alert codes and logs, see the Sun SeeBeyond eGate Integrator System Administration Guide.*

Chapter 3

Operating SSL

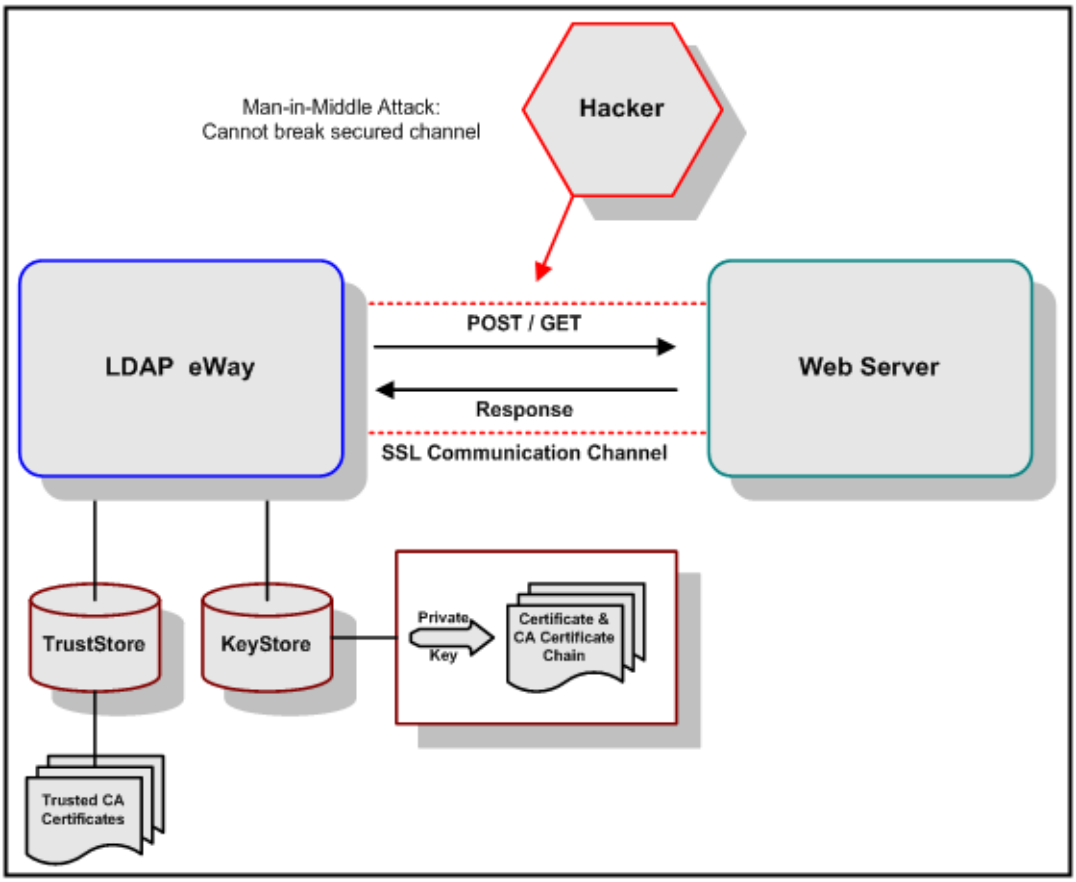
This chapter explains the operation of the Secure Sockets Layer (SSL) feature available with the LDAP eWay.

What's in This Chapter

3.1 Overview

The use of SSL with LDAP, enables LDAP data exchanges that are secure from unauthorized interception from "hackers" or other entities. The eWay's SSL feature provides a secure communications channel for the data exchanges (see Figure 4).

Figure 4 General SSL Operation: LDAP



This SSL feature is supported through the use of JSSE version 1.0.3.

Currently, the JSSE reference implementation is used. JSSE is a provider-based architecture, meaning that there is a set of standard interfaces for cryptographic algorithms, hashing algorithms, secured-socket-layered URL stream handlers, and so on.

Because the user is interacting with JSSE through these interfaces, the different components can be mixed and matched as long as the implementation is programmed under the published interfaces. However, some implementations may not support a particular algorithm.

The JSSE 1.0.3 application programming interface (API) is capable of supporting SSL versions 2.0 and 3.0 and Transport Layer Security (TLS) version 1.0. These security protocols encapsulate a normal bidirectional stream socket and the JSSE 1.0.3 API adds transparent support for authentication, encryption, and integrity protection. The JSSE reference implementation implements SSL version 3.0 and TLS 1.0.

For more information, visit the Sun Java Web site at the following URL:

http://java.sun.com

Note: *See the JSSE documentation provided by Sun Microsystems for further details.*

3.2 KeyStores and TrustStores

As depicted in Figure 4, JSSE makes use of files called *KeyStores* and *TrustStores*. The KeyStore is used by the eWay for client authentication, while the TrustStore is used to authenticate a server in SSL authentication.

- A *KeyStore* consists of a database containing a private key and an associated certificate, or an associated certificate chain. The certificate chain consists of the client certificate and one or more certification authority (CA) certificates.
- A *TrustStore* contains only the certificates trusted by the client (a "trust" store). These certificates are CA root certificates, that is, self-signed certificates. The installation of the Logical Host includes a TrustStore file named **cacerts.jks** in the location:

  `<c:\JavaCAPS>\logicalhost\is\domains\<MyDomain>\config`

  where `<c:\JavaCAPS>` is the directory where the Sun Java Composite Application Platform Suite is installed and `<MyDomain>` is the name of your domain. This file is recommended as the TrustStore for the LDAP eWay.

Both KeyStores and TrustStores are managed by means of a utility called **keytool**, which is a part of the Java SDK installation.

3.2.1 Generating a KeyStore and TrustStore

This section explains steps on how to create both a KeyStore and a TrustStore (or import a certificate into an existing TrustStore such as the default Logical Host TrustStore in the location:

```
<c:\JavaCAPS>\logicalhost\is\domains\<MyDomain>\config\cacerts.jks
```

where `<c:\JavaCAPS>` is the directory where the Sun Java Composite Application Platform Suite is installed and `<MyDomain>` is the name of your domain. The primary tool used is **keytool**, but **openssl** is also used as a reference for generating **pkcs12** KeyStores.

For more information on **openssl** and available downloads, visit the following Web site:

http://www.openssl.org.

3.2.2 KeyStores

This section explains how to use KeyStores.

Creating a KeyStore in JKS Format

This section explains how to create a KeyStore using the JKS format as the database format for both the private key, and the associated certificate or certificate chain. By default, as specified in the java.security file, **keytool** uses JKS as the format of the key and certificate databases (KeyStore and TrustStores). A CA must sign the certificate

signing request (CSR). The CA is therefore trusted by the server-side application to which the eWay is connected.

Note: *It is recommended to use the default KeyStore <c:\JavaCAPS>\logicalhost\is\domains\<MyDomain>\config\keystore.jks where <c:\JavaCAPS> is the directory where the Sun Java Composite Application Platform Suite is installed and <MyDomain> is the name of your domain.*

To generate a KeyStore

Use the following command:

```
keytool -keystore clientkeystore -genkey -alias client
```

You are prompted for several pieces of information required to generate a CSR. A sample key generation section follows:

```
Enter keystore password: seebyond
What is your first and last name?
[Unknown]: development.seebeyond.com
What is the name of your organizational unit?
[Unknown]: Development
what is the name of your organization?
[Unknown]: SeeBeyond
What is the name of your City of Locality?
[Unknown]: Monrovia
What is the name of your State or Province?
[Unknown]: California
What is the two-letter country code for this unit?
[Unknown]: US
Is<CN=Foo Bar, OU=Development, O=SeeBeyond, L=Monrovia,
ST=California, C=US> correct?
[no]: yes

Enter key password for <client>
    (RETURN if same as keystore password):
```

If the KeyStore password is specified, then the password must be provided for the eWay. Press RETURN when prompted for the key password (this action makes the key password the same as the KeyStore password).

This operation creates a KeyStore file **clientkeystore** in the current working directory. You must specify a fully-qualified domain for the "first and last name" question. The reason for this use is that some CAs such as Verisign expect this properties to be a fully qualified domain name.

There are CAs that do not require the fully qualified domain, but it is recommended to use the fully-qualified domain name for the sake of portability. All the other information given must be valid. If the information can not be validated, a CA such as Verisign does not sign a generated CSR for this entry.

This KeyStore contains an entry with an alias of **client**. This entry consists of the Generated private key and information needed for generating a CSR as follows:

```
keytool -keystore clientkeystore -certreq alias client -keyalg rsa
    -file client.csr
```

This command generates a certificate signing request which can be provided to a CA for a certificate request. The file **client.csr** contains the CSR in PEM format.

Some CA (one trusted by the Web server to which the eWay is connecting) must sign the CSR. The CA generates a certificate for the corresponding CSR and signs the certificate with its private key. For more information, visit the following web sites:

http://www.thawte.com

or

http://www.verisign.com

If the certificate is chained with the CA's certificate, perform step 1; otherwise, perform step 2 in the following list:

1. The following command assumes the client certificate is in the file **client.cer** and the CA's certificate is in the file **CARoot.cer**:

   ```
   keytool -import -keystore clientstore -file client.cer -alias
       client
   ```

   This command imports the certificate (which can include more than one CA in addition to the Client's certificate).

   Also use the following command to import the CA's certificate into the KeyStore for chaining with the client's certificate:

   ```
   keytool -import -keystore clientkeystore -file CARootcer -alias
       theCARoot
   ```

2. The following command imports the client's certificate signed by the CA whose certificate was imported in the preceding step:

   ```
   keytool -import -keystore clientkeystore -file client.cer -alias
       client
   ```

The generated file **clientkeystore** contains the client's private key and the associated certificate chain used for client authentication and signing. The KeyStore and/or **clientkeystore**, can then be used as the eWay's KeyStore.

See the **"KeyStores" on page 25** for more information.

Creating a KeyStore in PKCS12 Format

This section explains how to create a PKCS12 KeyStore to work with JSSE. In a real working environment, a customer could already have an existing private key and certificate (signed by a known CA). In this case, JKS format can not be used, because it does not allow the user to import/export the private key through **keytool**. It is necessary to generate a PKCS12 database consisting of the private key and its certificate.

The generated PKCS12 database can then be used as the eWay's KeyStore. The **keytool** utility is currently lacking the ability to write to a PKCS12 database. However, it can read from a PKCS12 database.

Note: *There are additional third-party tools available for generating PKCS12 certificates, if you want to use a different tool.*

For the following example, **openssl** is used to generate the PKCS12 KeyStore:

```
cat mykey.pem.txt mycertificate.pem.txt>mykeycertificate.pem.txt
```

The existing key is in the file **mykey.pem.txt** in PEM format. The certificate is in **mycertificate.pem.txt**, which is also in PEM format. A text file must be created which contains the key followed by the certificate as follows:

```
openssl pkcs12 -export -in mykeycertificate.pem.txt -out
    mykeystore.pkcs12 -name myAlias -noiter -nomaciter
```

This command prompts the user for a password. The password is required. The KeyStore fails to work with JSSE without a password. This password must also be supplied as the password for the eWay's KeyStore password (see **Table 7 on page 47**).

This command also uses the **openssl pkcs12** command to generate a PKCS12 KeyStore with the private key and certificate. The generated KeyStore is **mykeystore.pkcs12** with an entry specified by the **myAlias** alias. This entry contains the private key and the certificate provided by the **-in** argument. The **noiter** and **nomaciter** options must be specified to allow the generated KeyStore to be recognized properly by JSSE.

3.2.3 TrustStores

Creating a TrustStore

For demonstration purposes, suppose you have the following CAs that you trust: **firstCA.cert, secondCA.cert, thirdCA.cert**, located in the directory **C:\cascerts**. You can create a new TrustStore consisting of these three trusted certificates.

To create a new TrustStore

Use the following command:

```
keytool -import -file C:\cascerts\firstCA.cert -alias firstCA
    -keystore myTrustStore
```

You must enter this command two more times, but for the second and third entries, substitute **secondCA** and **thirdCA** for **firstCA**. Each of these command entries has the following purposes:

1. The first entry creates a KeyStore file name **myTrustStore** in the current working directory and imports the **firstCA** certificate into the TrustStore with an alias of **firstCA**. The format of **myTrustStore** is JKS.
2. For the second entry, substitute **secondCA** to import the **secondCA** certificate into the TrustStore, **myTrustStore**.
3. For the third entry, substitute **thirdCA** to import the **thirdCA** certificate into the TrustStore.

Once completed, myTrustStore is available to be used as the TrustStore for the eWay.

Using an Existing TrustStore

This section explains how to use an existing TrustStore such as the default Logical Host TrustStore in the location:

```
<c:\JavaCAPS>\logicalhost\is\domains\<MyDomain>\config\cacert
s.jks
```

where `<c:\JavaCAPS>` is the directory where the Sun Java Composite Application Platform Suite is installed and `<MyDomain>` is the name of your domain. The primary tool used is **keytool**, but **openssl** is also used as a reference for generating **pkcs12** KeyStores.

Notice that in the previous section, steps 2 and 3 were used to import two CAs into the TrustStore created in step 1. For example, suppose you have a trusted certificate file named: **C:\trustedcerts\foo.cert** and want to import it to the **trustedcacertsjks** TrustStore.

If you are importing certificates into an existing TrustStore, use:

```
keytool -import -file C:\cacerts\secondCA.cert -alias secondCA
    -keystore trustedcacertsjks
```

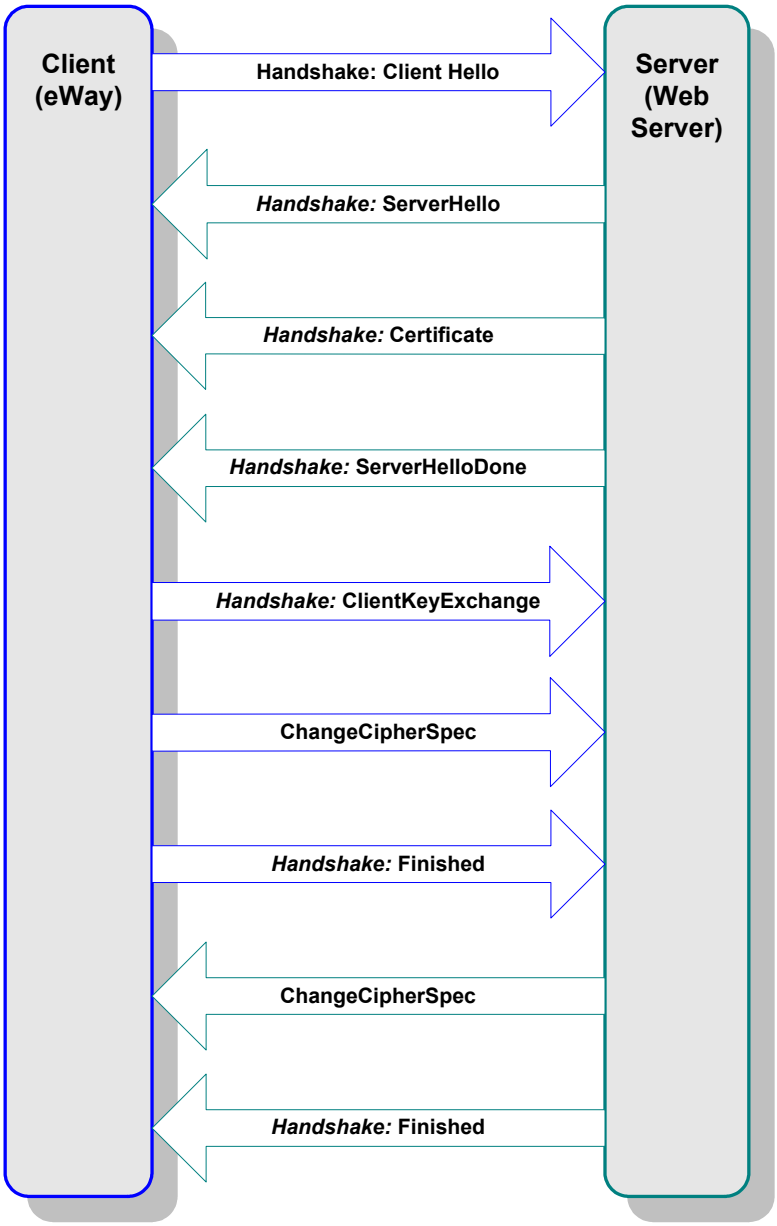
Once you are finished, **trustedcacertsjks** can be used as the TrustStore for the eWay.

3.3 SSL Handshaking

There are two options available for setting up SSL connectivity with a Web server:

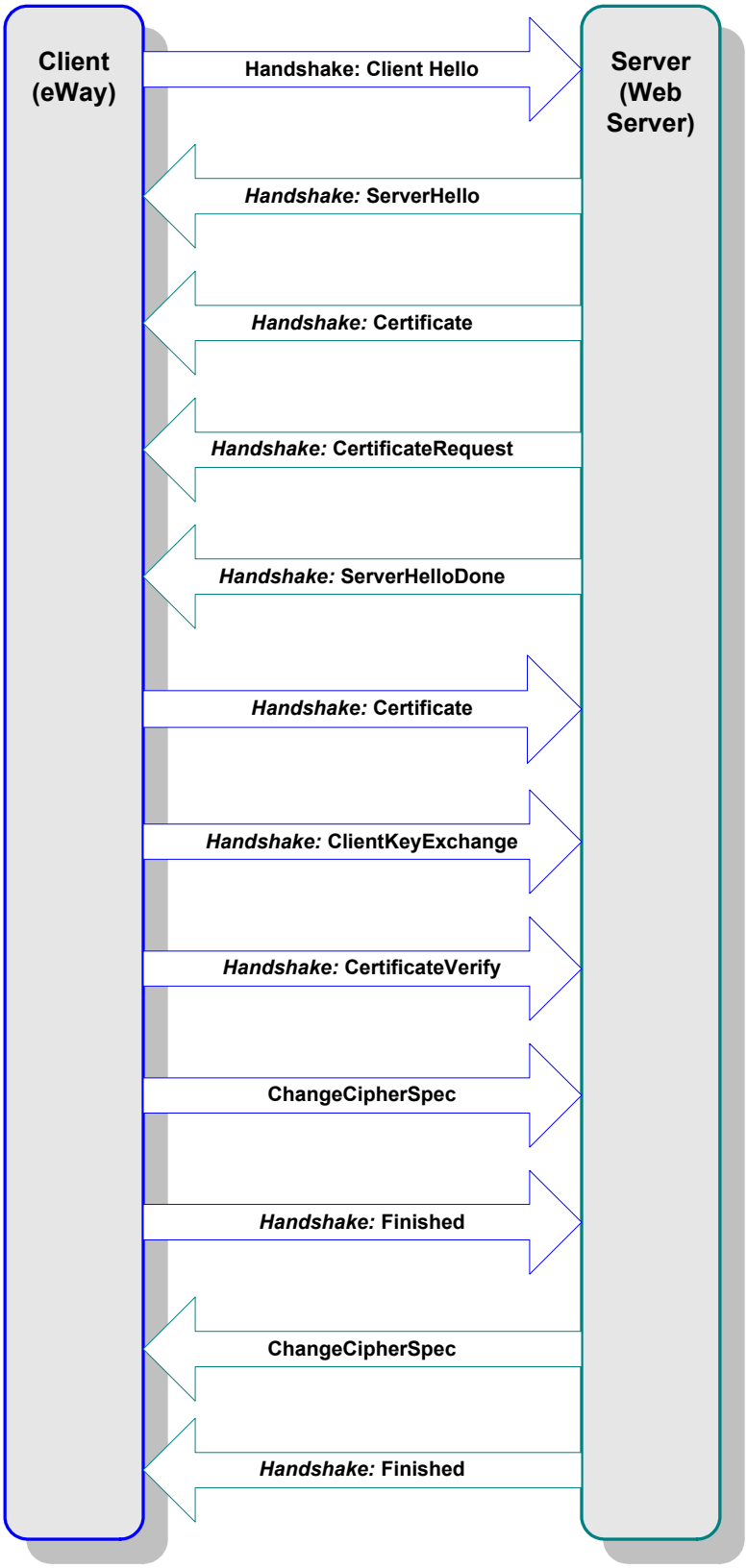
- **Server-side Authentication**: The majority of eCommerce Web sites on the Internet are configured for server-side authentication. The eWay requests a certificate from the Web server and authenticates the Web server by verifying that the certificate can be trusted. Essentially, the eWay performs this operation by looking into its TrustStore for a CA certificate with a public key that can validate the signature on the certificate received from the Web server. This option is illustrated in Figure 5.

Figure 5 Server-side Authentication



- **Dual authentication**: This option requires authentication from both the eWay and Web server. The server side (Web server) of the authentication process is the same as that described previously. In addition, however, the Web server requests a certificate from the eWay. The eWay then sends its certificate to the Web server. The server, in turn, authenticates the eWay by looking into its TrustStore for a matching trusted CA certificate. The communication channel is established by the process of both parties' requesting certificate information. This option is illustrated in Figure 6.

Figure 6 Dual Authentication



3.4 Using the OpenSSL Utility

The **OpenSSL** utility is a free implementation of cryptographic, hashing, and public key algorithms such as 3DES, SHA1, and RSA respectively. This utility has many options including certificate signing, which **keytool** does not provide. You can download **OpenSSL** from the following Web site:

http://www.openssl.org

Follow the build and installation instruction for **OpenSSL**.

To learn more about SSL, and the high level aspects of cryptography, a good source of reference is a book entitled *SSL and TLS: Designing and Building Secure Systems* (by Eric Rescorla, Published by Addison Wesley Professional; ISBN: 0201615983).

3.4.1 Creating a Sample CA Certificate

The sample given in this section demonstrates the use of the **OpenSSL** utility to create a CA. This generated CA is then used to sign a CSR (see **"Signing Certificates With Your Own CA" on page 33**), whether it is generated from **keytool** or **OpenSSL**.

For testing purposes a sample CA can be generated. To avoid spending additional funds to have a commercial CA sign test certificates, a sample is generated and used to sign the test certificate.

Perform the following operations from the command line:

```
openssl  req  -config c:\openssl\bin\openssl.cnf  -new  -x509  -
keyout  ca-key.pem.txt -out  ca-certificate.pem.txt  -days  365

Using properties from c:\openssl\bin\openssl.cnf
Loading 'screen' into random state: done
Generating a 1024 bit RSA private key
..................++++++
.....................++++++
writing new private key to 'ca-key.pem.txt'
Enter PEM pass phrase:
Verifying password: Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be
    incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name
    or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) []:US
State or Province Name (full name) []:California
Locality Name (eg, city) []:Monrovia
Organization Name (eg, company) []:SeeBeyond
Organizational Unit Name (eg, section) []:Development
Common Name (eg, your websites domain name) []
    :development.seebeyond.com
Email Address []:development@seebeyond.com
```

You are prompted for information. You must enter a password and remember this password for signing certificates with the CA's private key. This command creates a

private key and the corresponding certificate for the CA. The certificate is valid for 365 days starting from the date and time it was created.

The properties file **C:\openssl\bin\openssl.cnf** is needed for the **req** command. The default **config.cnf** file is in the **OpenSSL** package under the **apps** sub-directory.

Note: *That to use this file in Windows, you must change the paths to use double back-slashes. See* **"Windows OpenSSL.cnf File Example" on page 34** *for a complete* **Config.cnf** *file example, which is known to work in a Windows environment.*

3.4.2 Signing Certificates With Your Own CA

The example in this section shows how to create a CSR with **keytool** and generate a signed certificate for the CSR with the CA created in the previous section. The steps shown in this section, for generating a **KeyStore** and a CSR, were already explained under **"Creating a KeyStore in JKS Format" on page 25**.

Note: *No details are given here for the* **keytool** *commands. See* **"Creating a KeyStore in JKS Format" on page 25** *for more information.*

To create a CSR with keytool and generate a signed certificate for the CSR

1

```
keytool -keystore clientkeystore -genkey -alias client

Enter keystore password:  seebeyond
What is your first and last name?
[Unknown]:  development.seebeyond.com
What is the name of your organizational unit?
[Unknown]:  Development
What is the name of your organization?
[Unknown]:  SeeBeyond
What is the name of your City or Locality?
[Unknown]:  Monrovia
What is the name of your State or Province?
[Unknown]:  California
What is the two-letter country code for this unit?
[Unknown]:  US
Is <CN=Foo Bar, OU=Development, O=SeeBeyond, L=Monrovia, ST=Californi
a, C=US>   correct?
[no]:  yes

Enter key password for <client>
(RETURN if same as keystore password):
```

2

```
keytool  -keystore clientkeystore  -certreq  -alias client  -
keyalg rsa  -file client.csr
```

3

```
openssl  x509  -req  -CA
    ca-certificate.pem.txt  CAkey ca-key.pem.txt
    -in client.csr -out client.cer  -days 365  -CAcreateserial
```

This is how we create a signed certificate for the associated CSR. The option **-CAcreateserial** is needed if this is the first time the command is issued. It is used to

create an initial serial number file used for tracking certificate signing. This certificate will be valid for 365 days.

4

```
keytool -import -keystore clientkeystore -file client.cer
    -alias client

Enter keystore password: seebeyond
keytool error: java.lang.Exception: Failed to establish chain from
    reply
```

You get an exception because there is no certificate chain in the client certificate so we have to import the CA's certificate into the **KeyStore** first. You can then import the client.cer itself to form a certificate chain. You need the following steps:

5

```
keytool  -import  -keystore clientkeystore  -file CA
    ca-certificate.pem.txt  -alias theCARoot

Enter keystore password:  seebeyond
Owner: EmailAddress=development@seebeyond.com, CN=development.seebeyo
nd.com, OU=Development, O=SeeBeyond, L=Monrovia, ST=California, C=US
Issuer: EmailAddress=development@seebeyond.com, CN=development.seebey
ond.com,
OU=Development, O=SeeBeyond, L=Monrovia, ST=California, C=US
Serial number: 0
Valid from: Tue May 08 15:09:07 PDT 2001 until: Wed May 08
    15:09:07 PDT 2002
Certificate fingerprints:
MD5:  60:73:83:A0:7C:33:28:C3:D3:A4:35:A2:1E:34:87:F0
SHA1: C6:D0:C7:93:8E:A4:08:F8:38:BB:D4:11:03:C9:E6:CB:9C:D0:72:D0
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

6

```
keytool -import -keystore clientkeystore -file  client.cer -alias
    client

Enter keystore password: seebeyond
Certificate reply was installed in keystore
```

Now that we have a private key and an associating certificate chain in the **KeyStore clientkeystore**, we can use it as a **KeyStore** for client (eWay) authentication. The only warning is that the CA certificate must be imported into the trusted certificate store of the Web server to which you will be connecting. Moreover, the Web server must be configured for client authentication (**httpd.conf** for Apache, for example).

This appendix contains the contents of the **openssl.cnf file** that can be used on Windows. Be sure to make the appropriate changes to the directories.

3.4.3 Windows OpenSSL.cnf File Example

This section contains the contents of the **openssl.cnf file** that can be used on Windows. Be sure to make the appropriate changes to the directories.

```
#
# SSLeay example properties file.
```

```
# This is mostly being used for generation of certificate requests.
#

RANDFILE = .rnd

####################################################################
[ ca ]
default_ca= CA_default# The default ca section

####################################################################
[ CA_default ]

dir      = G:\\openssl\\\bin\\demoCA# Where everything is kept
certs    = $dir\\certs    # Where the issued certs are kept
crl_dir= $dir\\crl   # Where the issued crl are kept
database= $dir\\index.txt# database index file.
new_certs_dir= $dir\\newcerts# default place for new certs.

certificate= $dir\\cacert.pem    # The CA certificate
serial   = $dir\\serial    # The current serial number
crl      = $dir\\crl.pem    # The current CRL
private_key= $dir\\private\\cakey.pem   # The private key
RANDFILE= $dir\\private\\private.rnd # private random number file

x509_extensions= x509v3_extensions# The extentions to add to the cert
default_days= 365    # how long to certify for
default_crl_days= 30# how long before next CRL
default_md= md5      # which md to use.
preserve = no        # keep passed DN ordering

# A few difference way of specifying how similar the request should l
ook
# For type CA, the listed attributes must be the same, and the option
al
# and supplied fields are just that :-)
policy   = policy_match

# For the CA policy
[ policy_match ]
countryName  = match
stateOrProvinceName= match
organizationName= match
organizationalUnitName= optional
commonName   = supplied
emailAddress = optional

# For the 'anything' policy
# At this point in time, you must list all acceptable 'object'
# types.
[ policy_anything ]
countryName= optional
stateOrProvinceName= optional
localityName= optional
organizationName= optional
organizationalUnitName= optional
commonName   = supplied
emailAddress = optional

####################################################################
[ req ]
default_bits= 1024
default_keyfile = privkey.pem
distinguished_name= req_distinguished_name
attributes= req_attributes
```

```
[ req_distinguished_name ]
countryName  = Country Name (2 letter code)
countryName_min= 2
countryName_max= 2

stateOrProvinceName= State or Province Name (full name)

localityName = Locality Name (eg, city)

0.organizationName= Organization Name (eg, company)

organizationalUnitName= Organizational Unit Name (eg, section)

commonName   = Common Name (eg, your website's domain name)
commonName_max= 64

emailAddress = Email Address
emailAddress_max= 40

[ req_attributes ]
challengePassword= A challenge password
challengePassword_min= 4
challengePassword_max= 20

[ x509v3_extensions ]
```

Note: *The following copyright notices apply:*

Copyright © 1998-2001 The OpenSSL Project. All rights reserved.

Copyright © 1994-2002 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved. **http://www.w3.org/Consortium/Legal/**

Chapter 4

Setting LDAP eWay Properties

This chapter explains how to set the properties for the LDAP eWay.

What's in This Chapter

4.1 Creating and Configuring a LDAP eWay

All eWays contain a unique set of default configuration parameters. After the eWays are established and a LDAP External System is created in the Project's Environment, the eWay parameters are modified for your specific system. The LDAP eWay configuration parameters are modified from two locations:

- From the **Connectivity Map**—which contains parameters specific to the LDAP eWay, and may vary from other eWays (of the same type) in the Project.
- From the **Environment Explorer** tree—which contains global parameters that commonly apply to all eWays (of the same type) in the Project. Saved parameters are shared by all eWays in the LDAP External System Properties window.

Note: *In version 5.1.2 and above, the Connection and Security sections have been copied to the LDAP External System Environment properties window. If you are upgrading a 5.1.0 or 5.1.1 project, then refer to* **Java CAPS 5.1.0 and 5.1.1 Upgrade Procedures** *on page 19 for additional upgrade procedures. Before setting your configurations, also refer to* **Configuration Precedence** *on page 20.*

Note: *Properties set from the Collaboration will override the corresponding properties in the Connectivity Map configuration. Any properties that are not overridden retain their configured default settings.*

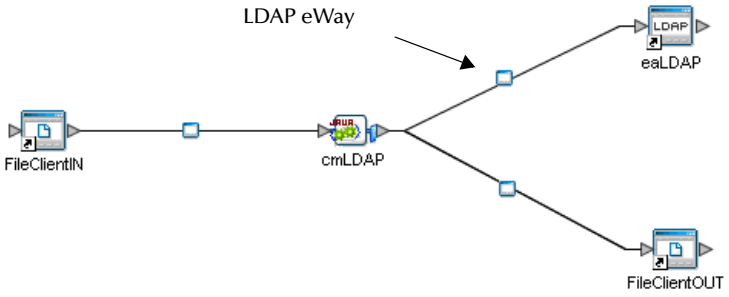
4.2 Configuring the eWay Connectivity Map Properties

When you connect an External Application to a Collaboration, Enterprise Designer automatically assigns the appropriate eWay to the link. Each eWay is supplied with a template containing default configuration properties that are accessible on the Connectivity Map.

To configure the eWay properties:

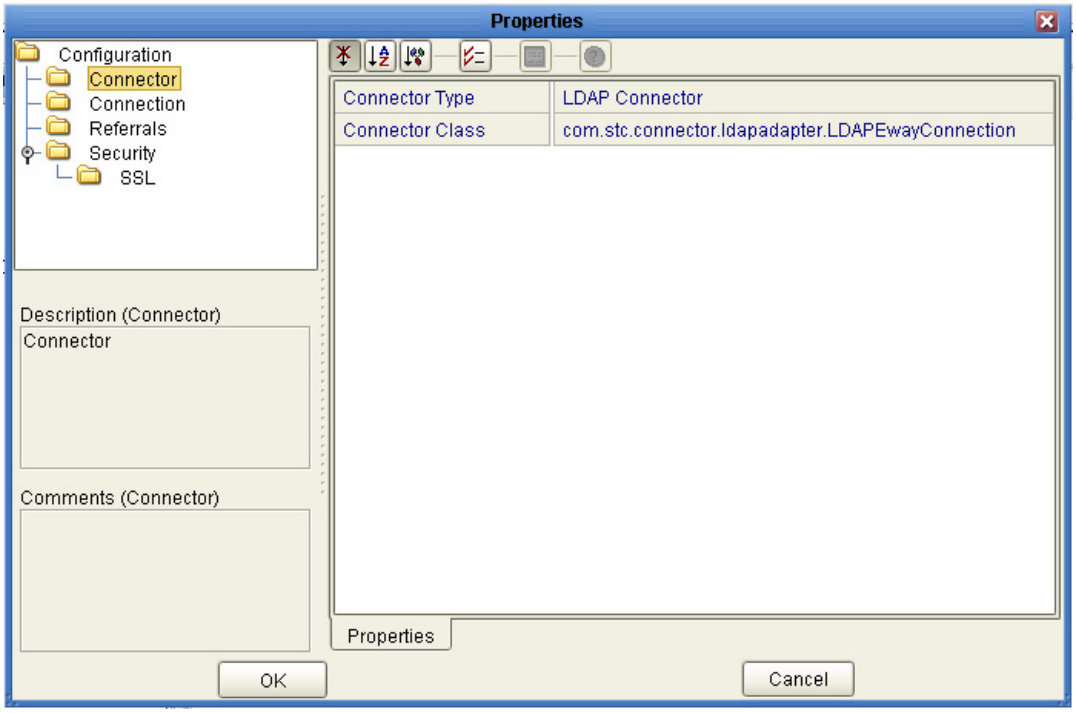
1 On the Enterprise Designer's Connectivity Map (see Figure 7), double-click the LDAP eWay icon. The Templates window appears.

Figure 7 Connectivity Map with Components



2 The Configuration properties window opens, displaying the default properties for the eWay.

Figure 8 Outbound eWay Properties



4.3 Configuring the eWay Environment Properties

The eWay Environment Configuration properties contain parameters that define how the eWay connects to and interacts with other eGate components within the Environment. When you create a new LDAP External System, you may configure the type of External System required.

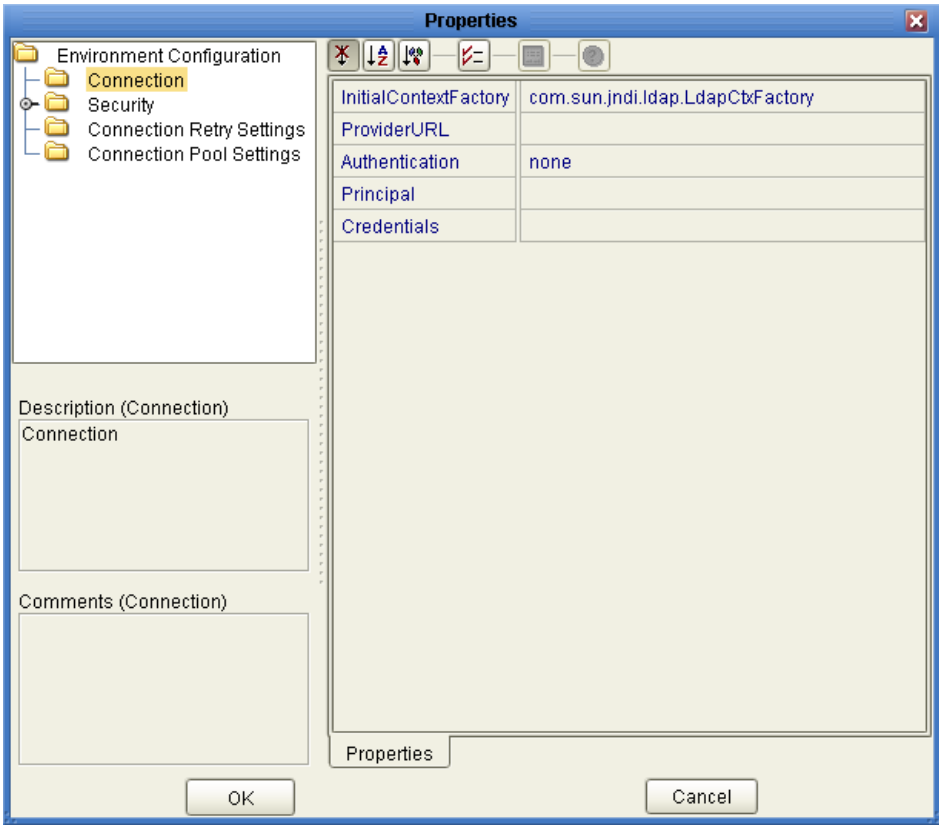
To Configure the Environment Properties:

1. In Enterprise Explorer, click the Environment Explorer tab.
2. Expand the Environment created for the LDAP Project and locate the LDAP External System.

Note: *For more information on creating an Environment, see the eGate Integrator Tutorial.*

3. Right-click the External System created for the LDAP Project and select Properties from the list box. The Environment Configuration Properties window appears.

Figure 9 LDAP eWay Environment Configuration



4 Click on any folder to display the default configuration properties for that section.

5 Click on any property field to make it editable.

After modifying the configuration properties, click **OK** to save the changes.

4.4 eWay Connectivity Map Properties

The eWay Connectivity Map consists of the following properties categories.

Outbound eWay Configuration Sections Include:

- **Configuring the Connector Section Properties** on page 40
- **Configuring Connection Section Properties** on page 41
- **Configuring the Referrals Section Properties** on page 42
- **Configuring the Security/SSL Section Properties** on page 46

4.4.1 Configuring the Connector Section Properties

The LDAP eWay Connector Section Properties include the following parameters:

Table 3 LDAP eWay — Connector Settings

Name	Description	Required Value
Connector Type	Lists the type of connector	The default is **LDAP Connector**.
Connector Class	Lists the Connector class.	The default connector class is com.stc.connector.ldapadapter.LDAPEwayConnection.

4.4.2 Configuring Connection Section Properties

The LDAP eWay Connection Section Properties allow you to define the connection to the LDAP system.

Note: *The Connection section in LDAP version 5.1.2 and above, has been copied to the Environment. If you are upgrading a project from version 5.1.0 or 5.1.1, please refer to the* **5.1.0 or 5.1.1 to 5.1.3 Upgrade Procedures** *on page 19.*

Table 4 LDAP eWay — Connection Settings

Name	Description	Required Value
Authentication	Allows you to select the authentication to be used (none or simple). Select the desired authentication as follows: ▪ **None:** No authentication, that is, an anonymous log-on. If you use this setting, ensure that the LDAP server supports anonymous log-ons. ▪ **Simple:** Authentication is based on a user name and password. You must provide the user name and password in the appropriate fields (Principal and Credentials).	Select **none** or **simple**; the default is **none**.
Credentials	Allows you to enter the credentials needed when using an authentication mechanism other than anonymous log-in (authentication = **none**).	The appropriate credentials, in the form of a valid password.
InitialContextFactory	Allows you to enter the factory to be used for creating the initial context for the LDAP server. By default the LDAP service provider provided by Sun, as part of the Java Software Developers' Kit (SDK), is used.	A valid Java factory name; the default is: com.sun.jndi.ldap.LdapCtxFactory. It is recommended that you do not change this value unless you want to use an LDAP service provider other than the one provided by Sun.

Table 4 LDAP eWay — Connection Settings

Name	Description	Required Value
Principal	Allows you to specify the principal needed when using an authentication mechanism other than anonymous log-in (authentication = **none**).	The fully qualified Distinguished Name (DN) of the user, for example: `CN=Administrator,CN=Users,DC=stc,dc=com`
ProviderURL	Allows you to specify the URL of the LDAP Server.	A valid URL with the protocol as **ldap**.

4.4.3 Configuring the Referrals Section Properties

The LDAP eWay Referrals Section Properties allow you to enter LDAP referral information.

Table 5 LDAP eWay — Referrals Settings

Name	Description	Required Value
Credentials	Allows you to specify the credentials file to be used when following any referrals in the directory. The credentials file is created using the RCF command-line utility.	A valid file and path name available to eGate.
Follow	Allows you to select whether referrals returned by an LDAP server must be followed.	Select **yes** or **no**; the default is yes. Enter the desired value as follows: ▪ **Yes:** Follow referrals. ▪ **No:** Referrals are not followed.

4.4.4 Additional Referrals Section Notes

Following are additional notes related to the properties found in the Referrals section.

Handling Search Referrals

A referral is an entity used to redirect a client's request to another server. A referral contains the names and locations of other objects. It is sent by the server to indicate that the information the client has requested can be found at another location (or locations), possibly at another server or several servers.

When you execute a search operation, you may encounter a referral entry, which is just a pointer to where that information can be found. The pointer is usually in a form similar to the **Provider URL** configuration of the eWay. It consists of the following components:

- Host name

- Port number
- Context name (optional)

You have the following options when you encounter a referral:

- **Ignore**: Ignore the referral.
- **Follow**: Follow the referral, that is, connect to the referred system and continue the search operation.
- **Throw**: Throw a referral exception, which can be caught by the client and action taken as needed.

With the LDAP eWay, you have the following properties you must set to work with referrals:

- **Credentials File**: Enter a fully qualified path to a file. This file must contain the appropriate referral credentials information (this file has to be generated using the RCF command line utility as explained later in this section).
- **Follow**: **Yes** or **No**. Default is **Yes**.

The scenarios shown in Table 6 can arise depending on the properties provided for the referrals and the behavior of the eWay, as explained for each of these scenarios.

Table 6 Referral Scenarios

Follow Setting	Credentials File	eWay Operation
Follow is set to **Yes**.	The credentials file is **not** provided.	The eWay uses the original credentials (user name and password) provided for the initial server and tries to connect to the referred system. The connection may fail if the referred system does not have the same credentials.
	The credentials file is provided and has the credentials entry for the referred host.	The connection to the initial server is configured to throw **LdapReferralException** when a referral is encountered which is subsequently caught by eWay. The eWay then establishes the connection to the referred system using the credentials information provided in the credentials file.
	The credentials file provided does **not** have the credentials entry for the referred host.	The connection to the initial server is configured to throw **LdapReferralException** when a referral is encountered, which is subsequently caught by the eWay. The eWay then establishes the connection to the referred system using an anonymous login. The connection may fail if the referred system does not allow an anonymous log-in.
Follow is set to **No**.	There is no credentials file.	Referrals are not followed, that is, the eWay ignores any referral.

To create a credentials file, you can use the Referral Credentials File (RCF) command-line utility.

Note: *Running the RCF utility on the command line without any parameters displays how to use the utility.*

To create a credentials file using the RCF utility:

1 The file to be used for the RCF utility are located at the following locations:

- <edesigner_home>\usrdir\modules\ext\ldapadapter\stcldap13.jar

or

- <edesigner_home>\usrdir\modules\ext\ldapadapter\stcldap14.jar

2 Copy and paste one of the above files to a folder and run the utility from this folder as follows:

```
<edesigner_home>\jdk\bin\java -cp ./stcldap13.jar
com.stc.connector.ldapadapter.utils.RCFUtil
```

The following menu displays:

```
C:\temp>java -cp ./stcldap13.jar
com.stc.connector.ldapadapter.utils.RCFUtil

Please specify the operation.

---+ RCFUtil +---

Interactive command line utility for creating and managing
file(s) containing credentials information to follow LDAP
referrals. File(s) generated can be used by the Java LDAP eWay
for following referrals that required credentials different
from those used to create the connection to the initial LDAP
server.

Usage : java com.stc.connector.ldapadapter.utils.RCFUtilOPTIONS
-- <filename>

OPTIONS:

--create Create a new referral credentials file.
--add Add an entry to the referral credentials file.
--list Print a list of entries in the referral credentials file.
--remove Remove an entry from the referral credentials file.
--modify Modify an entry in the referral credentials file.
--decrypt When displaying credentials, decrypt the credentials.
--username <username>  Specify the username; if not specified,
  it'll be prompted.
--password <password>  Specify the password; if not specified,
  it'll be prompted.
--help Print this usage.

filename:

The full path to the referral credentials file.
```

3 To create a new referral file called "samplercf.txt", enter the following parameters on the command line:

```
<edesigner_home>\jdk\bin\java -cp ./stcldap13.jar
com.stc.connector.ldapadapter.utils.RCFUtil --create -- samplercf.txt
```

This action requests a user name and password. Enter the user name and password. This user name and password is for protecting the file itself, because the file contains sensitive credential information about other LDAP servers. For example:

```
C:\temp>c:\ICAN510\edesigner\jdk\bin\java -cp .\stcldap13.jar
com.stc.connector.ldapadapter.utils.RCFUtil
--create -- samplercf.txt
Creating file samplercf.txt...
Enter username >> test
Enter password >> test
File created!
```

A message "File created!" appears. The file name here is samplercf.txt. The extension does not matter.

To add credentials information to the file:

1. To add LDAP Server connection info to a referral file called "samplercf.txt", enter the following parameters on the command line:

```
<edesigner_home>\jdk\bin\java -cp ./stcldap13.jar
com.stc.connector.ldapadapter.utils.RCFUtil --add --
samplercf.txt
```

2. Username and password are required to access the file. Provide the user name and password given for creating the file previously.
3. When the following prompts appear, enter the following information, as indicated:
   - Prompts for the host name: Enter the host name.
   - Prompts for the port number: Enter the LDAP port number.
   - Prompts for the principal: Enter the fully qualified DN of the user.
   - Prompts for the password: Enter the password for the DN specified previously.

   For example:

```
C:\temp>c:\ICAN510\edesigner\jdk\bin\java -cp .\stcldap13.jar
com.stc.connector.ldapadapter.utils.RCFUtil --add --
samplercf.txt
Adding a referral credentials entry...
Enter username >> test
Enter password >> test
Enter LDAP Host >> localhost.stc.com
Enter LDAP Port >> 389
Enter the Principal >> cn=Manager,dc=stc,dc=com
Enter the Credentials >> secret

Done.
```

To view the contents of the credentials file:

1. To view LDAP Server connection info in a referral file called "samplercf.txt", enter the following parameters on the command line:

```
<edesigner_home>\jdk\bin\java -cp ./stcldap13.jar
com.stc.connector.ldapadapter.utils.RCFUtil --list --
samplercf.txt
```

2. Username and password are required to access the file. Provide the user name and password given for creating the file previously.
3. The entries in the file are listed as shown in the following single-entry example:

```
1> localhost.stc.com | 389 | cn=Manager,dc=stc,dc=com | l/
ZRt1cfNKc=
```

4 The password is encrypted. To display the password in its decrypted form add --decrypt to the previous command. The output becomes:

```
1> localhost.stc.com | 389 | cn=Manager,dc=stc,dc=com | secret
```

For example:

```
C:\temp>c:\ICAN510\edesigner\jdk\bin\java -cp .\stcldap13.jar
com.stc.connector.ldapadapter.utils.RCFUtil --list --
samplercf.txt
Listing entries in the referral credentials file...
Enter username >> test
Enter password >> test
1> localhost.stc.com | 389 | cn=Manager,dc=stc,dc=com | l/
ZRt1cfNKc=

C:\temp>c:\ICAN510\edesigner\jdk\bin\java -cp .\stcldap13.jar
com.stc.connector.ldapadapter.utils.RCFUtil --list --decrypt --
samplercf.txt
Listing entries in the referral credentials file...
Enter username >> test
Enter password >> test
1> localhost.stc.com | 389 | cn=Manager,dc=stc,dc=com | secret
```

Other operations, such as removing a credential entry and modifying a credential entry for an entry, can be done using the RCF utility in the same way.

The following example shows the content of a credentials file, "samplercf.txt", with explanatory comments:

```
###This properties file was generated by
#com.stc.connector.ldapadapter.utils.RCFUtil.
#Do NOT modify this file "by hand" if you don't understand the
nature
#or format of this file. Use the utility to create and
#manage this file.
#
#Tue Feb 14 17:49:17 PST 2006
password=P9He6eCUY6Q\=
localhost.stc.com\:389=test;P9He6eCUY6Q\=
username=test
#New credentials entry that was created.
```

4.4.5 Configuring the Security/SSL Section Properties

The LDAP eWay Security/SSL Section Properties are used to set the basic security features for SSL.

Note: *The Security/SSL section from LDAP version 5.1.2 and above, has been copied to the Environment. If you are upgrading a project from version 5.1.0 or 5.1.1, please refer to the* **5.1.0 or 5.1.1 to 5.1.3 Upgrade Procedures** *on page 19.*

Table 7 LDAP eWay — Security/SSL Settings

Name	Description	Required Value
JSSE Provider Class	Specifies the fully qualified name of the JSSE provider class. For more information, see the Sun Microsystems Java site at: http://java.sun.com/	The name of a valid JSSE provider class; the default is: com.sun.net.ssl.internal.ssl.Provider If you are running the Integration Server on AIX, specify: com.ibm.jsse.IBMJSSEProvider
KeyStore	Specifies the default KeyStore file. The keystore is used for key/certificate management when establishing SSL connections.	A valid package location. There is no default value. It is recommended to use `<c:\JavaCAPS>\logicalhost\is\domains\<MyDomain>\config\keystore.jks` where `<c:\JavaCAPS>` is the directory where the Sun Java Composite Application Platform Suite is installed and `<MyDomain>` is the name of your domain.
KeyStore password	Specifies the default KeyStore password. The password is used to access the KeyStore used for key/certificate management when establishing SSL connections; there is no default.	A valid KeyStore password. There is no default value.
KeyStore type	Allows you to specify the default KeyStore type. The keystore type is used for key/certificate management when establishing SSL connections. If the KeyStore type is not specified, the default KeyStore type, JKS, is used.	A valid KeyStore type.
KeyStore username	The user name for accessing the keystore used for key/certificate management when establishing SSL connections. ***Note:*** If the keystore type is PKCS12 or JKS, the keystore user name property is not used. PKCS12 and JKS keystore types require passwords for access but do not require user names. If you enter a value for this property, it is ignored for PKCS12 and JKS.	A valid KeyStore user name.

Table 7 LDAP eWay — Security/SSL Settings

Name	Description	Required Value
SSL Connection Type	Allows you to specify the type of SSL connection to be used.	Select None, Enable SSL, or TLS On Demand. Enter the desired value as follows: **None:** No SSL, simple plain connection. **Enable SSL:** SSL is enabled. All communication to the LDAP server uses a secure communication channel. ***Note:*** If you are using the Enable SSL option, the ProviderURL property must point to a secure LDAP port (the default is 636). For additional information on required values for this property, see **SSL Connection Type** on page 49.
SSL Protocol	The SSL protocol to use when establishing an SSL connection with the LDAP server. See your JSSE documentation for information on your Logical Host's platform.	Select **TLS**, **TLSv1**, **SSLv3**, **SSLv2**, or **SSL**.
TrustStore	Specifies the default TrustStore. The TrustStore is used for CA certificate management when establishing SSL connections.	A valid TrustStore file; there is no default.
TrustStore password	Allows you to specify the default TrustStore password. The password is for accessing the TrustStore used for CA certificate management when establishing SSL connections.	A valid TrustStore password; there is no default.
TrustStore type	Allows you to specify the TrustStore type of the TrustStore used for CA certificate management when establishing an SSL connection. If the TrustStore type is not specified, the default TrustStore type, JKS, is used.	A valid TrustStore type.

Table 7 LDAP eWay — Security/SSL Settings

Name	Description	Required Value
Verify hostname	Determines whether the host name verification is done on the server certificate during the SSL handshake. You can use this property to enforce strict checking of the server host name in the request URL and the host name in the received server certificate.	**True** or **False**; the default is **False**. For additional information on required values for this property, see **Verify Hostname** on page 50.
X509 Algorithm Name	Specifies the X509 algorithm name to use for the trust and key manager factories.	The name of a valid **X509** algorithm; the default is SunX509. If you are running the Integration Server on AIX, specify **IbmX509**.

4.4.6 Additional Security/SSL Property Notes

Listed below are additional notes for the following Security/SSL section properties:

- **SSL Connection Type** on page 49
- **Verify Hostname** on page 50

SSL Connection Type

Make sure that the SSL properties, including security certificate installation, port number, and so on, are set correctly for the current LDAP server.

Transport Layer Security (TLS) is a protocol that guarantees privacy and data integrity between client/server applications communicating over the Internet. The TLS operation for this eWay supports both secure and nonsecure communication on the same connection.

However, some LDAP servers are required to start on a configured nonsecure port and cannot start on a secure port. For details, see the appropriate documentation for the LDAP server.

- **TLS on Demand**: A feature of LDAP version 3 (**StartTLS** extended operation), which is supported in Java SDK version 1.4 and later. Selecting this option allows you to establish an SSL connection on demand programmatically.

Note: *If you are using the* **TLS on Demand** *option, the* **ProviderURL** *property must point to a nonsecure LDAP port (the default is 389).*

After selecting this option, whenever secure communication is required, you must place any method call to the LDAP server between **startTLS** and **stopTLS** calls, which can be accessed via the LDAP OTD.

In the following example, the call to **performAddEntry** goes through a secure communication channel, but the call to **performRename** goes through a nonsecure plain-communication channel:

```
startTLS();
performAddEntry();
stopTLS();

performRename();
```

Make sure that the TLS settings (in addition to the SSL settings) are configured correctly for the current LDAP server.

Note: *Using the stopTLS method may cause unexpected behavior with some LDAP servers. You may need to remove the use of this method in your Collaboration Definitions. For example, you cannot use the stopTLS method when connecting to a Sun ONE Directory server. For details, see the appropriate documentation for the LDAP server.*

For information on how to use this feature with the LDAP OTD, see **TlsExtension Node** on page 77.

Verify Hostname

Under some circumstances, you can get different Java exceptions, depending on whether you set this property to **True** or **False**. This section explains what causes these exceptions.

For example, suppose the host name in the URL is **localhost**, and the host name in the server certificate is **localhost.stc.com**. Then, the following conditions apply:

- If **Verify hostname** is set to **False**:

  Host name checking between the requested URL and the server certificate is turned *off*.

  You can use an incomplete domain host name, for example, **https://localhost:444**, or a complete domain host name, for example, **https://localhost.stc.com:444**, and get a positive response in each case.

- If **Verify hostname** is set to **True**:

  Host name checking between the requested URL and the server certificate is turned *on*.

Note: *If you use an incomplete domain host name, for example, https://localhost:444, you can get the exception java.io.IOException: HTTPS hostname wrong.*

You must use a complete domain host name, for example, https://localhost.stc.com:444.

Note: *If the Java SDK version used by the Logical Host and the corresponding Logical Host property setting do not match, you can get the exception java.lang.ClassCastException.*

4.5 eWay External Properties

The eWay External System consists of the following properties categories.

4.5.1 Configuring Connection Section Properties

The LDAP eWay Connection Section Properties allow you to define the connection to the LDAP system.

Note: *The Connection section is new in LDAP version 5.1.3. If you are upgrading a project from version 5.1.0 or 5.1.1, please refer to the* **5.1.0 or 5.1.1 to 5.1.3 Upgrade Procedures** *on page 19.*

Table 8 LDAP eWay — Connection Settings

Name	Description	Required Value
Authentication	Allows you to select the authentication to be used (none or simple). Select the desired authentication as follows: ▪ **None:** No authentication, that is, an anonymous log-on. If you use this setting, ensure that the LDAP server supports anonymous log-ons. ▪ **Simple:** Authentication is based on a user name and password. You must provide the user name and password in the appropriate fields (Principal and Credentials).	Select **none** or **simple**; the default is **none**.
Credentials	Allows you to enter the credentials needed when using an authentication mechanism other than anonymous log-in (authentication = **none**).	The appropriate credentials, in the form of a valid password.

Table 8 LDAP eWay — Connection Settings

Name	Description	Required Value
InitialContextFactory	Allows you to enter the factory to be used for creating the initial context for the LDAP server. By default the LDAP service provider provided by Sun, as part of the Java Software Developers' Kit (SDK), is used.	A valid Java factory name; the default is: com.sun.jndi.ldap.LdapCtxFactory. It is recommended that you do not change this value unless you want to use an LDAP service provider other than the one provided by Sun.
Principal	Allows you to specify the principal needed when using an authentication mechanism other than anonymous log-in (authentication = **none**).	The fully qualified Distinguished Name (DN) of the user, for example: `CN=Administrator,CN=Users,DC=stc,dc=com`
ProviderURL	Allows you to specify the URL of the LDAP Server.	A valid URL with the protocol as **ldap**.

4.5.2 Configuring the Security/SSL Section Properties

The LDAP eWay Security/SSL Section Properties are used to set the basic security features for SSL.For more information on SSL Section properties, refer to **Additional Security/SSL Property Notes** on page 49

Note: *The Security/SSL section is new in LDAP version 5.1.3. If you are upgrading a project from version 5.1.0 or 5.1.1, please refer to the* **5.1.0 or 5.1.1 to 5.1.3 Upgrade Procedures** *on page 19.*

Table 9 LDAP eWay — Security/SSL Settings

Name	Description	Required Value
JSSE Provider Class	Specifies the fully qualified name of the JSSE provider class. For more information, see the Sun Microsystems Java site at: http://java.sun.com/	The name of a valid JSSE provider class; the default is: com.sun.net.ssl.internal.ssl.Provider If you are running the Integration Server on AIX, specify: com.ibm.jsse.IBMJSSEProvider
KeyStore	Specifies the default KeyStore file. The keystore is used for key/certificate management when establishing SSL connections.	A valid package location; there is no default.

Table 9 LDAP eWay — Security/SSL Settings

Name	Description	Required Value
KeyStore password	Specifies the default KeyStore password. The password is used to access the KeyStore used for key/certificate management when establishing SSL connections; there is no default.	A valid KeyStore password. There is no default value.
KeyStore type	Allows you to specify the default KeyStore type. The keystore type is used for key/certificate management when establishing SSL connections. If the KeyStore type is not specified, the default KeyStore type, JKS, is used.	A valid KeyStore type.
KeyStore username	The user name for accessing the keystore used for key/certificate management when establishing SSL connections. ***Note:*** If the keystore type is PKCS12 or JKS, the keystore user name property is not used. PKCS12 and JKS keystore types require passwords for access but do not require user names. If you enter a value for this property, it is ignored for PKCS12 and JKS.	A valid KeyStore user name.

Table 9 LDAP eWay — Security/SSL Settings

Name	Description	Required Value
SSL Connection Type	Allows you to specify the type of SSL connection to be used.	Select None, Enable SSL, or TLS On Demand. Enter the desired value as follows: **None:** No SSL, simple plain connection. **Enable SSL:** SSL is enabled. All communication to the LDAP server uses a secure communication channel. ***Note:*** If you are using the Enable SSL option, the ProviderURL property must point to a secure LDAP port (the default is 636). For additional information on required values for this property, see **SSL Connection Type** on page 49.
SSL Protocol	The SSL protocol to use when establishing an SSL connection with the LDAP server. See your JSSE documentation for information on your Logical Host's platform.	Select **TLS**, **TLSv1**, **SSLv3**, **SSLv2**, or **SSL**.
TrustStore	Specifies the default TrustStore. The TrustStore is used for CA certificate management when establishing SSL connections.	A valid TrustStore file; there is no default.
TrustStore password	Allows you to specify the default TrustStore password. The password is for accessing the TrustStore used for CA certificate management when establishing SSL connections.	A valid TrustStore password; there is no default.
TrustStore type	Allows you to specify the TrustStore type of the TrustStore used for CA certificate management when establishing an SSL connection. If the TrustStore type is not specified, the default TrustStore type, JKS, is used.	A valid TrustStore type.

Table 9 LDAP eWay — Security/SSL Settings

Name	Description	Required Value
Verify hostname	Determines whether the host name verification is done on the server certificate during the SSL handshake. You can use this property to enforce strict checking of the server host name in the request URL and the host name in the received server certificate.	**True** or **False**; the default is **False**. For additional information on required values for this property, see **Verify Hostname** on page 50.
X509 Algorithm Name	Specifies the X509 algorithm name to use for the trust and key manager factories.	The name of a valid **X509** algorithm; the default is SunX509. If you are running the Integration Server on AIX, specify **IbmX509**.

4.5.3 Configuring the Connection Retry Settings

The LDAP eWay Connection Retry Settings properties include the following parameters:

Table 10 LDAP External eWay Properties— Connection Retry Settings

Name	Description	Required Value
Maximum Retries	Maximum number of retries to establish a connection upon failure to acquire one.	There is no required value. The default value is **5**.
Retry Interval	The number of Milliseconds to wait between connection retries.	Any valid number. The default value is **10000**.

4.5.4 Configuring the Connection Pool Settings

The LDAP eWay Connection Pool Settings properties include the following parameters:

Table 11 LDAP External eWay Properties— Connection Pool Settings

Name	Description	Required Value
Steady Pool Size	The minimum number of connections that must be maintained in the pool.	The default value is **1**.
Maximum Pool Size	The maximum number of connections allowed in the pool. 0 (zero) indicates that there is no maximum.	The default value is **10**.
Maximum Idle Timeout	The maximum time in Seconds that a connection can remain idle in the pool. Zero indicates that there is no limit.	The default value is **300**.

Chapter 5

Using the LDAP OTD

This chapter explains the LDAP OTD that allows the LDAP eWay to communicate with LDAP. The chapter also gives details on the OTD node structure, including the nodes, available methods and properties, their applications, and how to use them.

What's in This Chapter

5.1 LDAP OTD Node Structure

This section explains the LDAP OTD node structure and layout, focusing on the subnodes of the OTD's root node. These subnodes and the sections that explain them are:

This section also explains subnodes and their features, under these nodes, where necessary.

5.1.1 Node Structure: Overview

The LDAP OTD (**LDAPClient**) exposes the Application Programming Interfaces (APIs) for accessing an LDAP directory in the eGate Java-based Collaboration environment. It is an uneditable read-only OTD.

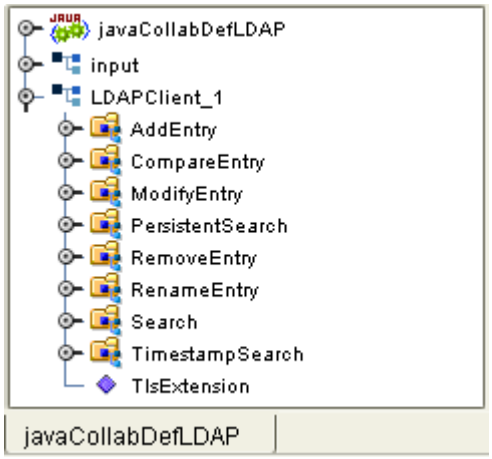
The LDAP OTD has the following components:

- The **LDAPClient** OTD itself, which exposes the structures and methods
- The Java classes, which implement those structures and methods

This chapter provides an overview of LDAP, then explains the LDAP OTD in detail, including how to use the LDAP OTD to build a Java-based Collaboration Definition for accessing an LDAP directory.

Figure 10 shows an example of the LDAP OTD in the eGate Enterprise Designer's Collaboration Editor (Java) window.

Figure 10 LDAP OTD in Enterprise Designer



The rest of the chapter provides the general outline of the OTD and the methods and properties exposed on each node. For a more detailed description of each method in the OTD, see the **Javadoc**.

5.1.2 LDAP Root Node

LDAPClient is the root node and provides a graphical representation of the interface. Expanding the node and each subnode reveals all the methods on the interface, which are themselves represented as nodes.

5.1.3 AddEntry Node

The **AddEntry** node is used to add entries to a directory. When adding an entry, there are different options available. To add an entry, specify the name of the entry to add (RDN relative to the initial context), the attributes and values for each attribute, and then call the **performAddEntry** method.

Figure 11 shows the **AddEntry** node in its expanded form.

Figure 11 AddEntry Node

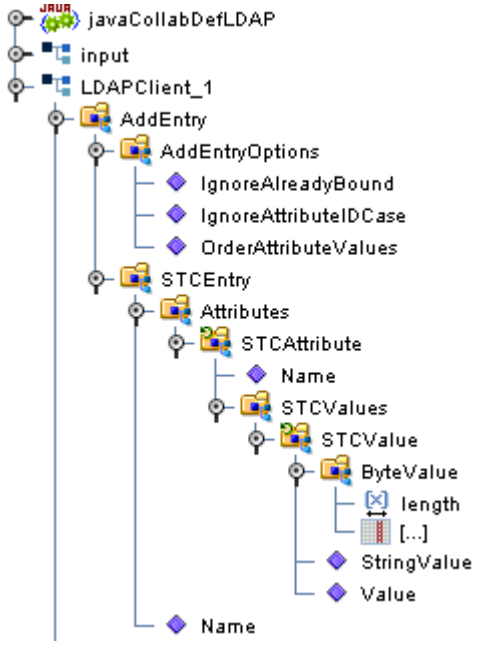


Table 12 explains the nodes and fields exposed on the **AddEntry** Node.

Table 12 AddEntry Node

Name	Description
AddEntryOptions node	Used to add entries to a directory and contains options used when adding an entry.
IgnoreAlreadyBound field	Set to **true** to ignore an **AlreadyBoundException** exception to be thrown if the entry to be added already exists in the directory. Set this field to **false** to force the eWay to throw an **LDAPApplicationException** when adding an existing entry. The same exception is thrown if any other internal errors have occurred.
IgnoreAttributeIDCase field	Tells the eWay to ignore the case-sensitivity of the attribute IDs (names) that are defined. This field is of type Boolean; set this field to **true** to ignore case-sensitivity or **false** to *not* ignore case-sensitivity. The default is **true**.
OrderAttributeValues field	Tells the eWay to order the values for each attribute. This field is of type Boolean; set this field to **true** to order the values of each attribute or **false** to ignore the order of the values. The default is **false**.
STCEntry node	See Table 13.

5.1.4 STCEntry Subnode

The **STCEntry** subnode appears many times in the LDAP OTD. This node has only two nodes under it, the **Name** and **Attributes**. The **Attributes** node contains a collection of attributes under the **STCAttribute** node.

This subnode is the basic container used to send data to the LDAP server. The structure of the **STCEntry** subnode is shown in Figure 11.

See Table 13 for a description of this subnode, showing the subnode levels under the **STCEntry** subnode.

Table 13 STCEntry Subnode

First Level	Second Level	Third Level	Fourth Level	Fifth Level	Description
Name					Name of the entry.
Attributes					Collection of attributes for the entry.
	STCAttribute				Container to hold the details of a single attribute.
		Name			Name of the attribute.
		STCValues			Collection of values for the attribute.
			STCValue		Container to hold a single value of the attribute's value.
				ByteValue	Value of the attribute as a byte array.
				StringValue	Value of the attribute as a string.
				Value	Value of the attribute as an object.

5.1.5 CompareEntry Node

You can use the **CompareEntry** to check for any existing attribute that has one or more specified values. To compare an entry, you specify the RDN of the entry to compare and the search filter for the comparison.

You can then invoke the **performCompare** method that returns **true** if the specified entry has any matching attribute with the values specified in the filter.

Figure 12 shows the **CompareEntry** node in its expanded form.

Figure 12 CompareEntry Node

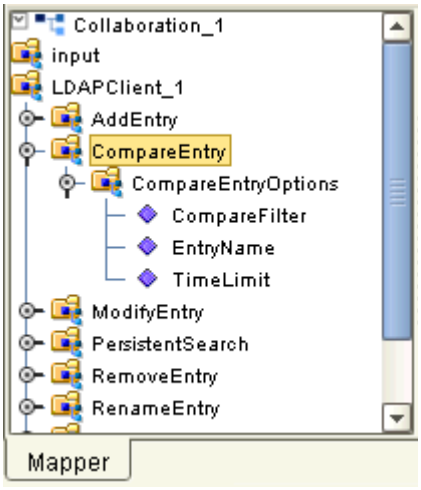


Table 14 explains the nodes and fields exposed on the **CompareEntry** node.

Table 14 CompareEntry Node

Name	Description
CompareEntryOptions node	Used to check for the existence of any attribute that has any value.
CompareFilter field	Consists of the attribute(s) and value(s) to search. **Example**: If **EntryName** is `"cn=John Doe, ou=People, dc=acme, dc=com"` and the **CompareFilter** is `"(password=jdoepassword)"`, then **performCompare** returns **true** if the specified entry, `"cn=John Doe, ou=People, dc=acme, dc=com"`, has an attribute called **password** whose value is equal to **jdoepassword**. If the specified entry does not have matching attributes and values, then **peformCompare** returns **false**. An **LDAPApplicationException** exception is thrown if there were other internal errors. **Example**: Comparing an entry that does not exist in the directory results in an **LDAPApplicationException** exception thrown because of a **NameNotFoundException** internal exception.
EntryName field	The DN of the entry to compare.
TimeLimit field	Used to specify the time-out, in milliseconds, for a compare operation. If the operation exceeds the set time limit, **performCompare** returns without results

5.1.6 ModifyEntry Node

You can use the **ModifyEntry** node to modify an existing entry in a directory. You can make the following modifications to an entry:

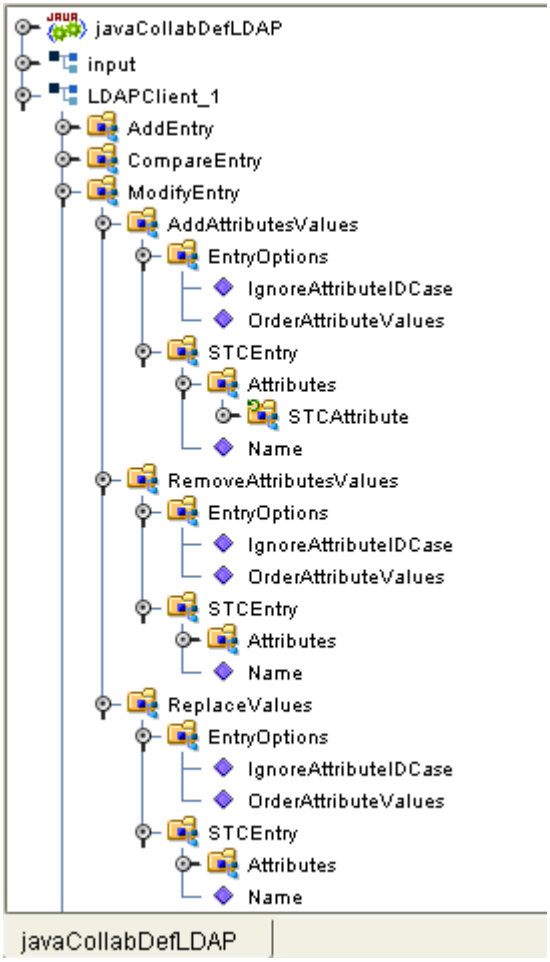
- Add any attribute to an entry.
- Add any value to any attribute of an entry.

- Remove any attribute from an entry.
- Remove any value from any attribute of an entry.
- Replace all existing values of any attribute with any new value for an entry.

Adding attributes and values is accomplished by using the **ModifyEntry** subnode called **AddAttributesValues**. Removing attributes and values is accomplished by using the **ModifyEntry** subnode called **RemoveAttributesValues**. Replacing values is accomplished by using **ModifyEntry** subnode called **ReplaceValues**.

Figure 13 shows the **ModifyEntry** node in its expanded form.

Figure 13 ModifyEntry Node



In Figure 13, the subnodes under the **STCEntry** nodes are identical in form to the subnodes under **STCEntry** in the **AddEntry** node. They also function in the same way. See Table 13 for details on these subnodes.

Table 15 explains the nodes and fields exposed on the **ModifyEntry** node.

Table 15 ModifyEntry Node

Name	Description
AddAttributesValues node	Used to specify the entry to modify and the attributes or values you want to add to the specified entry. To add attributes or values to an entry, specify the options, the name of the entry to modify, the attributes and values, and call the **performAddAttributesValues** method. If the modification is successful, **performAddAttributesValues** returns **true**. Otherwise, an **LDAPApplicationException** exception is thrown or **false** is returned. **How It Works:** If values are specified for an attribute, and the attribute does not exist for the entry, the attribute and values are added for that entry. If values are specified for an attribute, and the attribute does exist for the entry, only the values are added to the attribute. An **LDAPApplicationException** exception is thrown if no value(s) are specified for an attribute. ***Note:*** Attempting to add an existing value results in an **AttributeInUseException** exception.
RemoveAttributesValues node	Used to specify the entry to modify and the attributes or values you want to remove from the specified entry. To remove attributes or values from an entry, specify the options, the name of the entry to modify, the attributes and values, and call the **performRemoveAttributesValues** method. If the modification is successful, **performRemoveAttributesValues** returns **true**. Otherwise, an **LDAPApplicationException** exception is thrown or **false** is returned. **How It Works:** If values are specified for an attribute, the values are removed. If all the values of the attribute are removed, the attribute itself is also removed. To remove an attribute, do not specify any values for that attribute. Instead, specify the attribute name you want to remove. ***Note:*** Attempting to remove a nonexistent value or attribute results in **NoSuchAttributeException** exception.

Table 15 ModifyEntry Node (Continued)

Name	Description
ReplaceValues node	Used to specify the entry to modify and the values for each of the attributes you want to replace. To replace the values of an attribute for an entry, specify the options, the name of the entry to modify, the attribute and values, and call the **performReplaceValues** method. If the modification is successful, **performReplaceValues** returns **true**. Otherwise, an **LDAPApplicationException** exception is thrown or **false** is returned. All existing values of an attribute are replaced by the newly specified values.
EntryOptions node	Used to specify options when you are *adding or removing* attributes and/or values to/from an entry. The following options can be set: **IgnoreAttributeIDCase** This field tells the eWay to ignore the case-sensitivity of the defined attribute IDs (names). This field is of type Boolean; set this field to **true** to ignore case-sensitivity or **false** to NOT ignore case-sensitivity. The default value is **true**. **OrderAttributeValues** This field tells the eWay to order the values for each attribute. It is of type Boolean. Set this field to **true** to order the values of each attribute or **false** to ignore the order of the values. The default value is **false**.
STCEntry node	See Table 13.
performAddAttributesValues method	Adds attributes and/or values. See **"AddAttributesValues node"**. To access, right-click on the **AddAttributesValues** node and select this method from the pop-up box.
performRemoveAttributesValues method	Removes attributes and/or values. See **"RemoveAttributesValues node"**. To access, right-click on the **RemoveAttributesValues** node and select the method from the pop-up box.
performReplaceValues method	Replaces values of an attribute. See **"ReplaceValues node"**. To access, right-click on the **ReplaceValues** node and select the method from the pop-up box.

5.1.7 PersistentSearch Node

The **PersistentSearch** node allows you to use the LDAP Persistent Search feature. Persistent Search is a control supported by LDAP version 3. This control lets you track updates on the LDAP server.

Note: *For an explanation of how to use the eWay to track LDAP updates using versions without Persistent Search, see* **TimestampSearch Node** *on page 76.*

Figure 14 shows the **PersistentSearch** node in its expanded form.

Figure 14 PersistentSearch Node

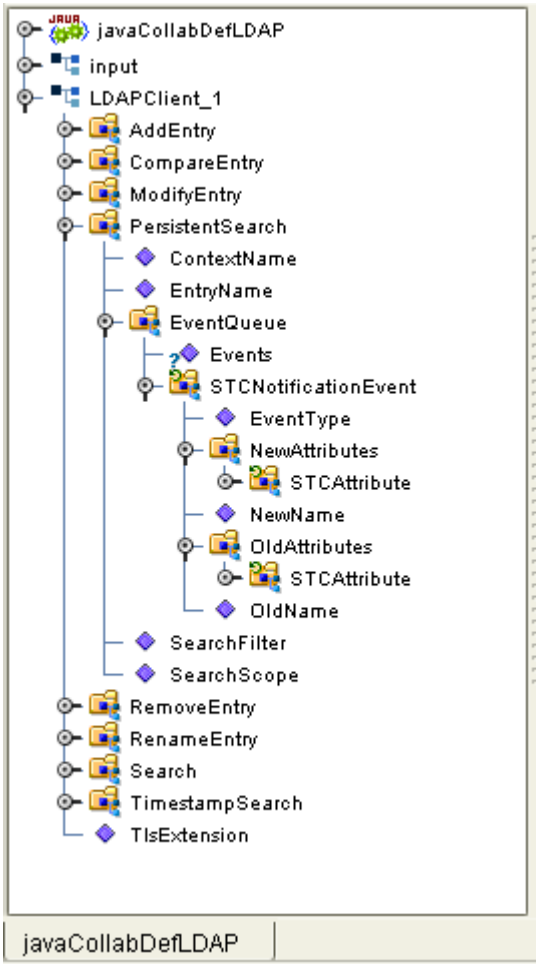


Figure 14 shows that the subnodes under **NewAttributes** and **OldAttributes** are identical in form to the subnodes under the **STCEntry** subnode. They also function in the same way. See Table 13 for details on these subnodes.

Persistent Search Limitations

The Persistent Search control feature has the following limitations:

- Works only against the Sun ONE server as OpenLDAP and Active Directory (on Windows 2000. Windows .NET supports this control) does not support persistent search control.

  On Sun ONE, when an object is renamed or removed, only the old object name is returned by Persistent Search. The Sun ONE Directory server does not return the attributes of the removed object or the attributes of the old object before renaming. This problem is a shortcoming in the server.

- Windows 2000 does not support Persistent Search control. Windows .NET, however, does support this control.

LDAP Version 3 Controls and Extensions

LDAP version 3 provides ways of extending functionality via controls, special operations like Persistent Search control, or LDAP extensions, that is, a user-specified extended functionality. Not all LDAP servers support these features. Earlier versions do not support them at all, and version 3 only supports a given control or extension if it has been implemented.

Before using a control or extension, be sure to find out whether the LDAP server supports the control or extension being used. In addition, do not enable a particular control/extension if the LDAP server does not support it. Doing so causes the eWay to fail with an **LDAPApplicationException** exception.

When you specify any control or extension, the LDAP eWay first checks whether the server supports the specified feature. If the server supports that control or extension, the requested operation is executed. If the control or extension is not supported, the eWay throws the exception along with a message specifying that the specified control or extension is not supported.

Using Persistent Search

To use Persistent Search, right-click on the **PersistentSearch** node and select the **search** method from the pop-up box. Calling this method initiates a Persistent Search operation.

The **search** method registers a listener on the LDAP server and keeps listening to events occurring on the server. These events are modifications, for example, adding an object, removing an object, renaming an object, or changing an object.

The retrieved events are stored in a queue you can access using the **EventQueue** node. This node contains the results of the current Persistent Search as an **STCNotificationEvent** node object. See **STCNotificationEvent Nodes** on page 78 for details on this node.

The **PersistentSearch** node consists of subnodes as shown in Table 16.

Table 16 PersistentSearch Node

Name	Description
ContextName field	Used to set the root of the search in the directory. The context name is relative to the context specified in the eWay's **ProviderURL** property. If the context name is not set correctly, the eWay is not able to properly resolve the context name relative to the initial eWay connection. **Example** (of a context name): `ou=MyOrg`, where `ou=MyOrg` is relative to **ldap://myldapserver1:389/dc=acme,dc=com**. In this case, `ou=MyOrg,dc=acme,dc=com` is the DN.
EntryName field	The name of the entry (RDN relative to the context name) on which the listener is registered.

Table 16 PersistentSearch Node (Continued)

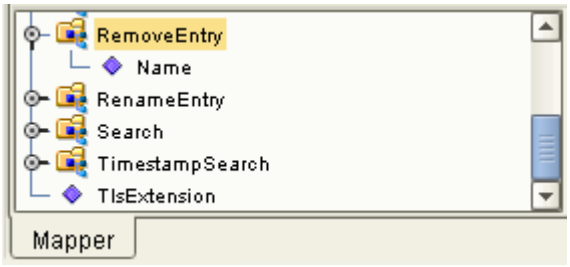
Name	Description
EventQueue node	This node contains the results returned as an **STCNotificationEvent** node object (see Table 24 for details on this node's structure).
SearchFilter field	The search filter expression per RFC 2254 to filter out the search results.
SearchScope field	The scope or boundary of the search. See **SearchOptions Scopes** on page 71 for details.

5.1.8 RemoveEntry Node

The **RemoveEntry** node can be used to remove an entry from the directory. To remove an entry, specify the RDN of the entry to remove and call the **performRemove** method.

Figure 15 shows the **RemoveEntry** node in its expanded form.

Figure 15 RemoveEntry Node



The **RemoveEntry** node has the **Name** field. You can set the name of the entry to remove in the **Name** field. Once the name is set, call the **performRemove** method (using the pop-up box) to remove the specified entry from the directory.

If the specified entry does not exist in the directory, a **NameNotFoundException** is thrown. An exception also occurs if any other internal error happens.

5.1.9 RenameEntry Node

The **RenameEntry** node can be used to rename an existing entry with a new name. To rename an entry, the user specifies the RDN of the entry to rename, the new RDN of the entry, and call the **performRename** method. The parent context specified by the new RDN must already exist.

For example, if the old RDN is **cn=John Doe, ou=People** and the new RDN is **cn=John Doe, ou=Staff**, then the parent context, **ou=Staff**, must already exist in the directory or else **performRename** fails with an exception.

Figure 16 shows the **RenameEntry** node in its expanded form.

Figure 16 RenameEntry Node

Table 17 describes the fields exposed on the **RenameEntry** node.

Table 17 RenameEntry Node

Field Name	Description
OldName	The entry's existing name. Before calling the **performRename** method, set the **OldName** field.
NewName	The entry's new name. Before calling the **performRename** method, set the **NewName** field.

Upon successfully renaming the entry, the **performRename** method returns **true**; otherwise **false** is returned.

An exception is thrown if any other internal errors have occurred. For example, renaming an entry that does not exist in the directory results in an **LDAPApplicationException** exception because of a **NameNotFoundException** internal exception.

5.1.10 Search Node

The **Search** node is specific to operations that are done once the eWay is connected to the LDAP server. The **Search** node corresponds to performing searches for an entry or multiple entries of the LDAP directory.

To perform a search, you specify the name context or starting entry for the search, the search scope or the boundaries to which the search is limited, and some search criteria known as a search filter.

The **Search** node, its subnodes, and fields are explained in the rest of this section. Figure 17 shows the **Search** node in its expanded form.

Figure 17 Search Node

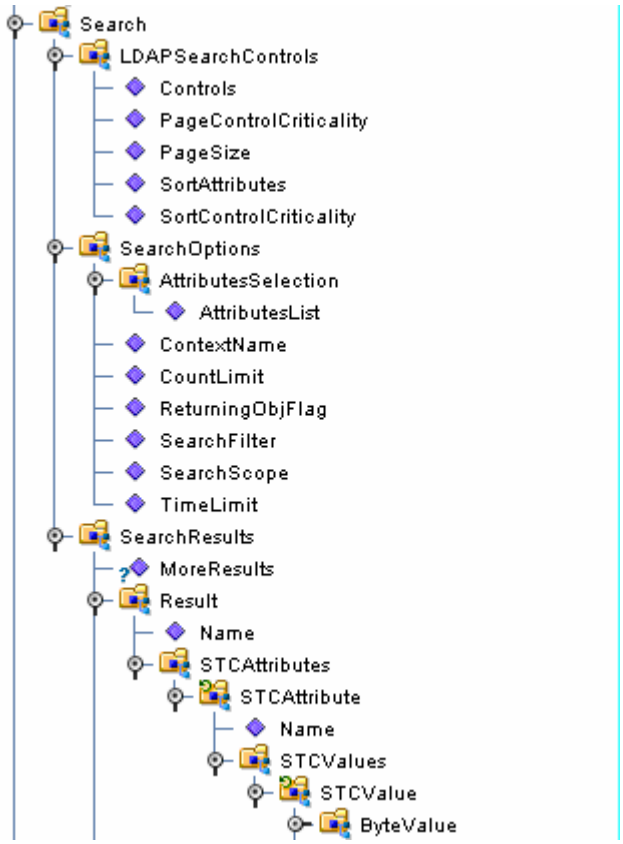


Figure 17 shows that the subnodes under **Result** are identical in form to the subnodes under the **STCEntry** subnode. They also function in the same way. See **Table 13 on page 59** for details on these subnodes.

The **Search** node has the following subnodes:

- **LDAPSearchControls**
- **SearchOptions**
- **SearchResults**

It has the following method:

- **performSearch**

To perform a search, first specify any LDAP search controls to use, then the search options such as the search filter, and finally call the **performSearch** method. Upon successfully calling the **performSearch** method, you retrieve the results of the search by using the **SearchResults** node.

LDAPSearchControls

LDAP version 3 provides a way of extending functionality through the use of controls.

Note: *Not all LDAP servers support specific controls or extensions. See* **LDAP Version 3 Controls and Extensions** *on page 65 for details.*

Table 18 explains the fields exposed on the **LDAPSearchControls** node.

Table 18 LDAPSearchControls Node

Field Name	Description
Controls	Contains a collection of controls that have been set.
PageControlCriticality	Used to set the criticality of the **PagedResultsControl**. The control can be set to **true** (critical) or **false** (noncritical). A critical control cannot be ignored by the server. In other words, if the server receives a critical control that it does not support, regardless of whether the control makes sense for the operation, if the operation is not performed, an **OperationNotSupportedException** is thrown.
PageSize	Allows you to specify the number of entries to return in a page.
SortAttributes	Allows you to request that the results returned be sorted according to the attributes specified. To use sort control, set the **SortAttributes** field with a string consisting of attributes, each separated by a pipe "|" character. **Example**: To sort entries returned by the attribute **cn** followed by the attribute **mail**, set **SortAttributes** with the string **cn|mail**.
SortControlCriticality	Allows you to set the criticality of the **SortControl**. The control can be set to **true** (critical) or **false** (noncritical). A critical control cannot be ignored by the server. In other words, if the server receives a critical control that it does not support, regardless of whether the control makes sense for the operation, if the operation is not performed, an **OperationNotSupportedException** is thrown.

Note: *Microsoft Active Directory requires the PagedResultsControl or else only a maximum of 1000 entries are returned, even if there are more than 1000 entries.*

Once the controls are set, subsequent searches send the control information to the server. To remove the controls, use the **removeSortControlAttributes** or **removePagedResultsControl** method (from the pop-up box on the **LDAPSearchControls** node). After a control is removed, subsequent searches do not send the information for removed control to the server.

SearchOptions

The **SearchOptions** node specifies the search criteria, such as the scope of the search and the search filter.

AttributesSelection

Under the **SearchOptions** node, the **AttributesSelection** node is used to restrict which attributes can be returned on a search. **AttributesSelection** is a collection of attribute IDs (names) you can manage using the following methods:

- **AddAttribute** takes an attribute name, as an argument, of type **java.lang.String**.
- **RemoveAttribute** also takes an attribute name as an argument which is of type **java.lang.String**.
- **ClearAttributes** does not take any arguments and removes any attributes added.

SearchOptions Node Fields

Table 19 explains the fields exposed on the **SearchOptions** node.

Table 19 SearchOptions Node

Field Name	Description
ContextName	Used to set the root of the search in the directory. The context name is relative to the context specified in the eWay's **ProviderURL** property. If the context name is not set correctly, the eWay is not able to properly resolve the context name relative to the initial connection. **Example** (of a context name): `ou=MyOrg`, where `ou=MyOrg` is relative to **ldap://myldapserver1:389/dc=acme,dc=com**. In this case, `ou=MyOrg,dc=acme,dc=com` is the DN.
CountLimit	Defines the maximum number of entries that can be returned on a search result.
SearchFilter	Used to specify the search filter for the search and is of type String. The basic search syntax is: `(Attribute Operator Value)` Where: ▪ **Attribute** is one of the possible attributes that an entry may have. ▪ **Operator** defines the comparison value, such as '='. ▪ **Value** is a value that an attribute may have. **Example**: `(cn=John Doe)` In the example, **cn** is the attribute, **=** is the operator, and **John Doe** is the value. The search filter specifies for entries where the attribute **cn** is equal to **John Doe**. For more information, see **SearchFilter Binary Operators** on page 72 and **SearchFilter Boolean Operators** on page 73. ***Note:*** The search filter syntax is described by RFC 2254. For more information, refer to this RFC on **http://www.ietf.org**.
SearchScope	The scope or boundary of the search. See **SearchOptions Scopes** on page 71 for details.
TimeLimit	Used to specify the time-out in milliseconds for a search. If the search exceeds the set time limit, **performSearch** returns without results.
returnObjFlag	Used to set the boolean flag to enable/disable returning objects returned as part of the result.

SearchOptions Scopes

This section explains the scope parameters **OBJECT_SCOPE, ONELEVEL_SCOPE,** and **SUBTREE_SCOPE.** Each figure shows a dotted box highlighting the scope and the entries covered for that scope parameter.

To specify the scope of the search, enter one of the values shown in Table 20 for the appropriate OTD field node.

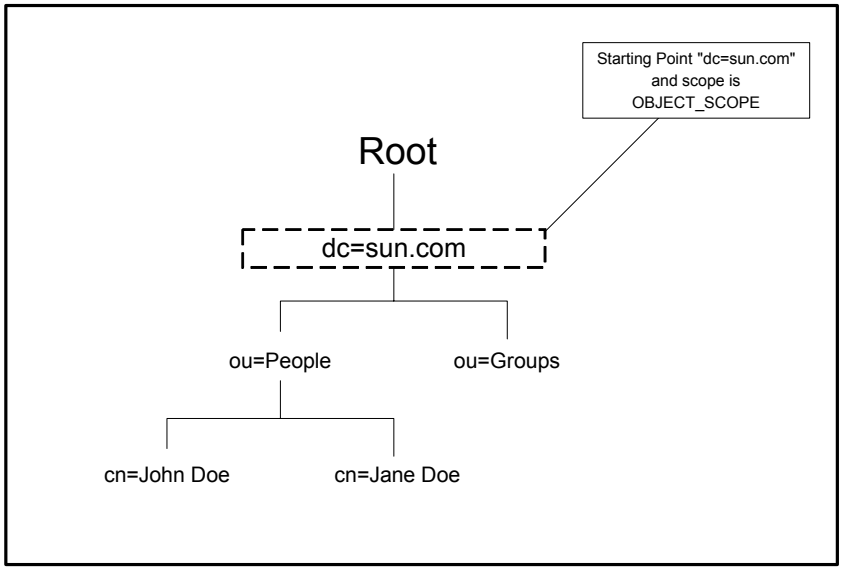
Table 20 Search Options Scope Parameters

Value	Parameter
0	**OBJECT_SCOPE**
1	**ONELEVEL_SCOPE**
2	**SUBTREE_SCOPE**

The following list explains these parameters:

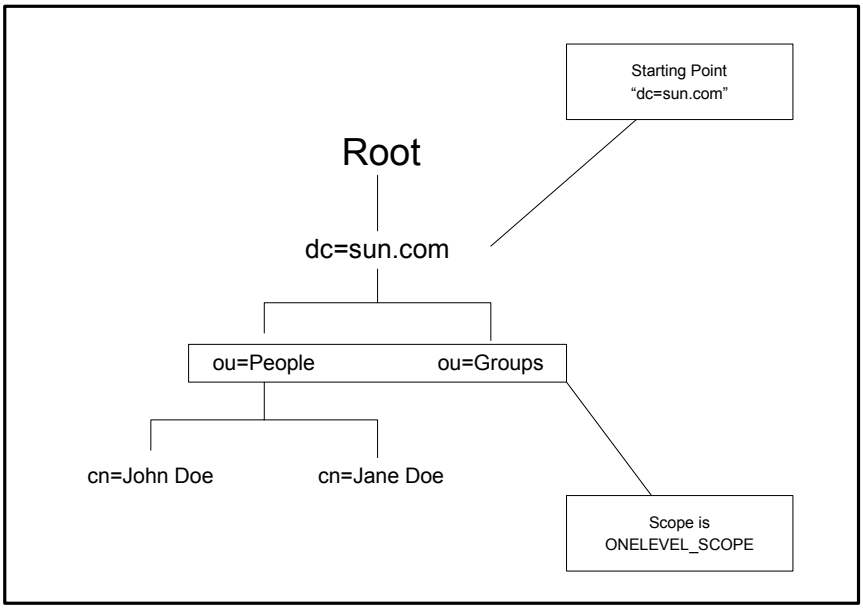
- **OBJECT_SCOPE** tells the eWay to search only within the named object, defined with **ContextName**. Using this scope essentially compares the named object for some particular attribute and/or value. See Figure 18.

Figure 18 OBJECT_SCOPE Diagram



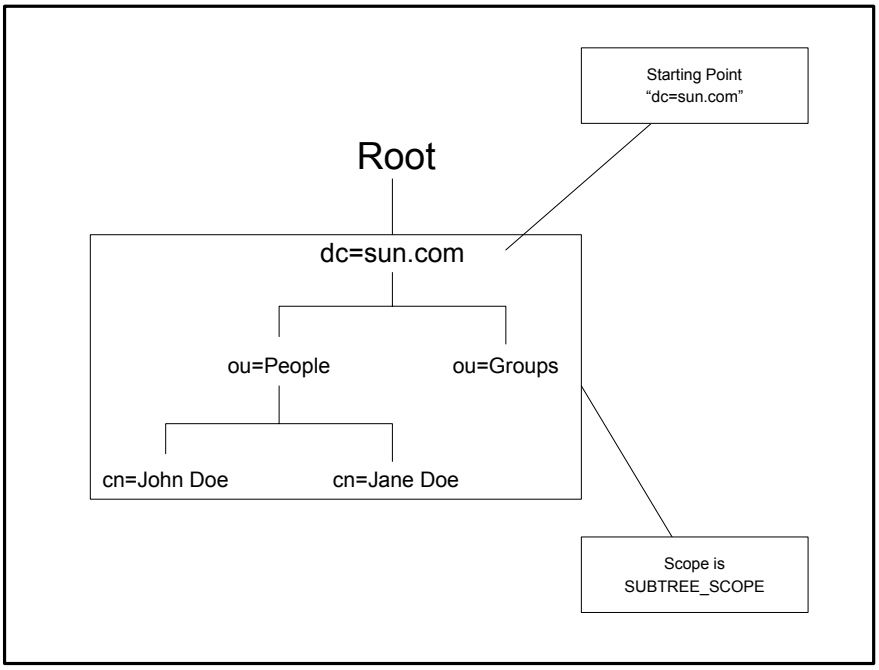
- **ONELEVEL_SCOPE** tells the eWay to search for entries one level below the named object. See Figure 19.

Figure 19 ONELEVEL_SCOPE Diagram



- **SUBTREE_SCOPE** tells the eWay to search for all entries starting from the named object and all descendants below the named object. See Figure 20.

Figure 20 SUBTREE_SCOPE Diagram



SearchFilter Binary Operators

Binary operators that can be used in a filter expression are listed in Table 21.

Note: *Not all servers support all the operators described in this guide. See your LDAP server administrator for information on which search operators are supported by your system.*

Table 21 Search Filter Binary Operators

Operator	Comments	Example
=	Retrieves entries that have a particular attribute equaling the specified value.	**(sn=Doe)** retrieves the entry with the attribute **sn**, which equals **Doe**.
>=	Retrieves entries that have a particular attribute whose value is greater than or equal to the specified value.	**(sn>=Doe)** retrieves all the entries whose attribute sn falls between **sn=Doe** and **sn=Z** … (D is less than Z).
<=	Retrieves entries that have a particular attribute whose value is less than or equal to the specified value.	**(sn<=Doe)** retrieves all the entries whose attribute **sn** falls between **sn=A …** and **sn=Doe** (A is less than D).
=*	Retrieves entries that have a particular attribute that has any value.	**(sn=*)** retrieves all the entries whose attribute **sn** has some value.
~=	Retrieves entries that have a particular attribute with some value similar to the specified value. This operator is used for approximate matches.	**(sn~=Doa)** retrieves the entry **sn=Doe**. **Doa** matches approximately to **Doe**.

SearchFilter Boolean Operators

You can define different conditions using binary operators combined with Boolean operators. The syntax for using Boolean operators is:

```
(Boolean_operator (filter) (filter)…(filter))
```

In this example of syntax, **filter** is an expression using one of the binary operators and the **Boolean_operator** is one of the following symbols: &, |, !. For example, **(| (cn=John Doe)(sn=Smith))**, gets the entries with attribute "**cn**" equal to **John Doe** or entries with attribute **sn** equal to **Smith**.

Boolean operators that can be used in a filter expression are listed in Table 22.

Table 22 Search Filter Boolean Operators

Operator	Comments	Example
&	Retrieves all entries that match all the search filter criteria.	**(& (sn=Smith)(telephoneNumber=444-4444))** retrieves entries with **sn** equal to **Smith** and **telephoneNumber** equal to **444-4444**.
|	Retrieves entries that match one or more of the search filter criteria.	**(|(sn=Smith)(sn=Doe))** retrieves entries with **sn** equal to **Smith** or **sn** equal to **Doe**.
!	Retrieves entries that do *not* match the search filter criteria. Only one search filter can be specified (that is, **(! (filter))** is allowed but **(!(filter)(filter))** is *not* allowed).	**(! (sn=Smith))** retrieves all entries with the attribute **sn** not equal to **Smith**.

Important: *If* **AddAttributesSelection** *is not used, all attributes are returned by default.*

SearchResults

The **SearchResults** node enables you to retrieve the results returned by the search. This node has the following methods:

- **nextResult**
- **hasResults**
- **hasMoreResults**
- **getNextResult**

After **performSearch** has been called, the resultant entries are stored internally for retrieval in **SearchResults**.

To determine whether results were returned from a search

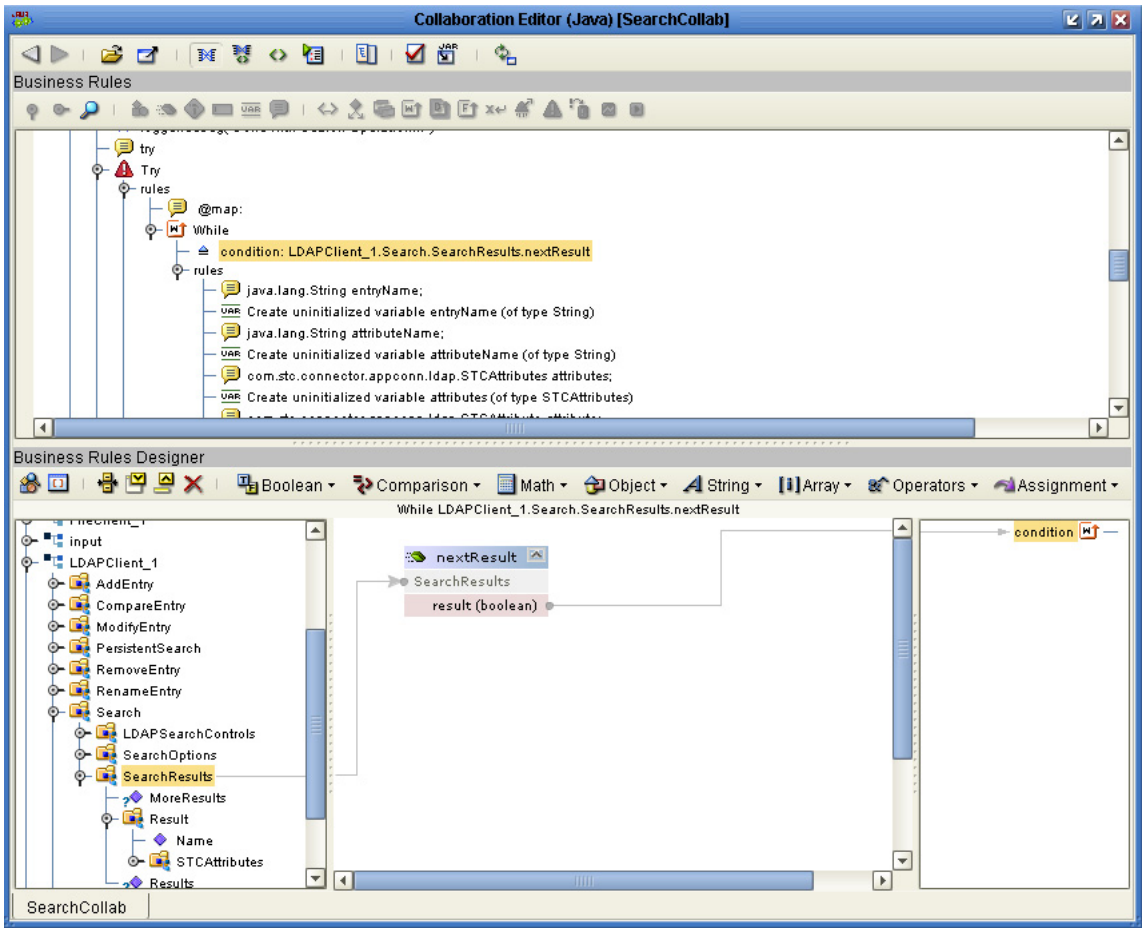
1. Call the **hasResults** method, which returns **true** if any results are returned, or **false** otherwise.
2. To iterate through all the entries, call **hasMoreResults** and **nextResult** within a **while** loop.

SearchResults operates as follows:

- The call to **hasResults** determines whether there are results and uses **hasMoreResults** as the condition for a **while** loop.
- Within the **while** loop, a call to **nextResult** populates the **Result** node with the next resultant entry.
- Once **nextResult** is called, the **Result** object is accessed.

See Figure 21 for a Java-based Collaboration Definition from a Project sample that illustrates the use of the **SearchResults** operation.

Figure 21 SearchResults Operation in Collaboration Editor (Java)



The following sample code displays the name of the result with the Java code:

```
System.out.println ("Entry>>>> " + LDAPClient_1.getSearch().getSea
rchResults().getResult().getName());
```

Result

As already explained, calling **nextResult** populates **Result** with the next result. The **Result** node has the **Name** field, of type **java.lang.String**, which holds the DN of the entry.

Result has the **STCAttributes** node, which is a collection of attributes. To determine the number of **STCAttribute** nodes, call the **countSTCAttribute** method, which returns an integer.

STCAttribute and STCValue

See Table 13 for details on these subnodes.

Retrieving Values for Attributes

Use the methods shown in the following lines of Java code to retrieve a value for an attribute:

```
if(LDAPClient_1.getSearch().getSearchResults().getResult().getSTCA
ttribute(i).getSTCValue(j).isString())
```

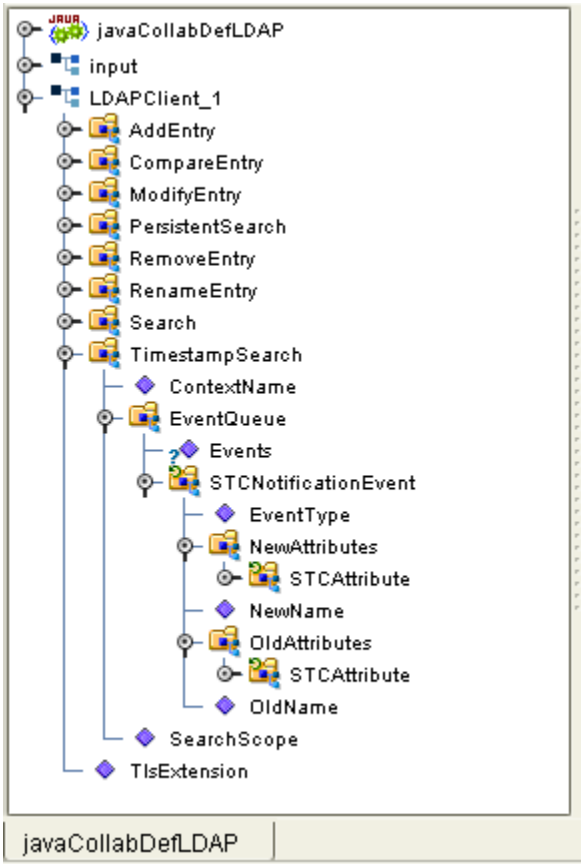
```
            {
               System.out.println ("    Value [String]: " +
getLDAP().getSearch().getSearchResults().getResult().
getSTCAttribute(i).getSTCValue(j).getStringValue());
            }
```

5.1.11 TimestampSearch Node

The **TimestampSearch** node allows you to track updates on an LDAP server that does not support Persistent Search control (see **PersistentSearch Node** on page 63). The node's operation uses the timestamp of the entries to track the updates.

Figure 22 shows the **TimestampSearch** node in its expanded form.

Figure 22 TimestampSearch Node



Timestamp Search Limitations

The Timestamp Search feature has the following limitations:

- Does not notify about the removal of objects, because this mechanism depends on the timestamp of the entry.
- Does not work against Active Directory, because Active Directory uses the local time zone instead of the standard GMT time zone.

- Does not work on the OpenLDAP server, because newly created objects are visible only after restarting the server. This problem is a shortcoming in the server.

Using Timestamp Search

To use Timestamp Search, right-click on the **TimestampSearch** node and select the **search** method from the pop-up box from within an infinite loop. The resulting search records the current time and begins comparing the timestamps of all the entries in the directory. Any entry whose timestamp is greater than the recorded timestamp is returned as a result.

The results are stored in a queue you can access using the **EventQueue** node. This node contains the results of the current search as an **STCNotificationEvent** node object. See **STCNotificationEvent Nodes** on page 78 for details on this node.

The **TimestampSearch** node consists of subnodes as shown in Table 23.

Table 23 TimestampSearch Node

Name	Description
ContextName field	Used to set the root of the search in the directory. The context name is relative to the context specified in the eWay's **ProviderURL** property. If the context name is not set correctly, the eWay is not able to properly resolve the context name relative to the initial eWay connection. **Example** (of a context name): `ou=MyOrg`, where `ou=MyOrg` is relative to **ldap://myldapserver1:389/dc=acme,dc=com**. In this case, `ou=MyOrg,dc=acme,dc=com` is the DN.
SearchScope	The scope or boundary of the search. See **SearchOptions Scopes** on page 71 for details.
EventQueue node	This node contains the results returned as an **STCNotificationEvent** object (see Table 24 for details on this node's structure).

5.1.12 TlsExtension Node

The **TlsExtension** node allows you to start and stop a **StartTLS** extended operation. The **StartTLS** extended operation is a feature of LDAP version 3, which is supported in the Java Software Developer's Kit (SDK) version 1.4 and later.

This feature allows you to establish an SSL connection on demand programmatically. To use this feature, you must call the LDAP server between **startTLS** and **stopTLS** method calls. You can access these methods by right-clicking the **TlsExtension** node and selecting the desired method from the pop-up box.

To enable this option, you must also set the **TLS On Demand** eWay property (**Security/SSL/SSL Connection Type**). See **SSL Connection Type** on page 49 for details and an example.

5.1.13 STCNotificationEvent Nodes

This node is used as a container for the results returned by the **PersistentSearch** and **TimestampSearch** operations.

The **STCNotificationEvent** nodes consist of subnodes as shown in Table 24.

Table 24 STCNotificationEvent Nodes

Name	Description
EventType	The type of event returned as the current result. The values can be: ▪ 0 - Object added (a new object was added to the directory) ▪ 1 - Object removed (an object was removed from the directory) ▪ 2 - Object renamed (an object was renamed) ▪ 3 - Object changed (an object was modified)
OldName	The name of the entry before the current operation. This could be null if the event type is **object added**.
NewName	The name of the entry after the current operation. This could be null if the event type is **object removed**.
OldAttributes	The attributes of the entry before the current operation. This could be null if the event type is **object added**.
NewAttributes	The attributes of the entry after the current operation. This could be null if the event type is **object removed**.

Note: *For more information on OTD nodes and methods, see the eGate Integrator User's Guide. For more information on Java classes and methods, see the Javadoc.*

Chapter 6

Reviewing the Sample Project

This section describes how LDAP eWay components are created and implemented in a Java Composite Application Platform Suite Project.

It is assumed that the reader understands the basics of creating a Project using the Enterprise Designer. For more information on creating an eGate Project, see the "*Sun SeeBeyond eGate™ Tutorial*" and the "*Sun SeeBeyond eGate™ Integrator User's Guide*".

What's in This Chapter

6.1 Sample Project Description

The LDAP sample Project demonstrates how the LDAP eWay processes information from a LDAP system. The resulting information is then written to a text file.

The **LDAP_eWay_Sample.zip** file contains sample Project data that provides basic instruction on creating a Collaborations that retrieves search results from XML based input files.

The ZIP file contains the following:

- input_data folder
- LDAP_SampleProject_510.zip

6.1.1 input_data Folder

When you extract the sample Project file, you get an additional folder called **input_data**. This folder contains the following:

- **LDAP Operations Folders**

  The **input_data** folder contains a set of folders named for the types of LDAP operations (add, remove, and so on). The folder for each operation contains at least one **DTD** file (used for the input data format) and multiple **XML** files.

- **Directory Structure Files**

  The **input_data** folder also contains a number of directory structure files required to run the LDAP Project. For additional information, see **Sample Project Directory Structure** on page 83.

A listing of files found in the input_data folder is seen in Table 25 below.

Table 25 input_data Folder Items

Folder	Type	Files
AddEntry	Operations Folder	input_ldapadd_AD.xml input_ldapadd_OL.xml input_ldapadd_SO.xml ldapadd.dtd
CompareEntry	Operations Folder	input_ldapcompare_AD.xml input_ldapcompare_OL.xml input_ldapcompare_SO.xml ldapcompare.dtd
LDIF > Active Directory	Directory structure folder	ldap_AD.ldif ldap_AD_adduser.ldif
LDIF > OpenLDAP	Directory structure folder	ldap_openldap.ldif
LDIF > SunOne	Directory structure folder	ldap_sunone.ldif
ModifyEntry	Operations Folder	ldapmodify.dtd input_ldapmodify_addattr_AD.xml input_ldapmodify_addattr_OL.xml input_ldapmodify_addattr_SO.xml input_ldapmodify_removeattr_AD.xml input_ldapmodify_removeattr_OL.xml input_ldapmodify_removeattr_SO.xml input_ldapmodify_replaceattr_AD.xml input_ldapmodify_replaceattr_OL.xml input_ldapmodify_replaceattr_SO.xml
PersistentSearch	Operations Folder	input_ldappersistentsearch_SO.xml ldappersistentsearch.dtd
RemoveEntry	Operations Folder	input_ldapremove_AD.xml input_ldapremove_OL.xml input_ldapremove_SO.xml ldapremove.dtd
RenameEntry	Operations Folder	input_ldaprename_AD.xml input_ldaprename_OL.xml input_ldaprename_SO.xml ldaprename.dtd
SearchEntry	Operations Folder	input_ldapsearch_AD.xml input_ldapsearch_OL.xml input_ldapsearch_SO.xml ldapsearch.dtd
TimestampSearch	Operations Folder	input_timestampsearch_SO.xml ldaptimestampsearch.dtd

6.1.2 LDAP_SampleProject_510.zip

The **LDAP_SampleProject_510** has been created for testing the following features of the LDAP eWay. These LDAP operations appear as Collaborations in your imported sample Project. The sample documented in this guide describes how the **SearchCollab** operation uses the import data format of the **ldapsearch.dtd** to pull values from an XML file.

Java Collaborations included with the **LDAP_SampleProject_510** sample Project are listed in Table 26.

Table 26 Java Collaborations in the LDAP_SampleProject_510

Java Collaborations	Purpose
AddCollab	Designed to add a new entry into the directory.
SearchCollab	Designed to search the directory for specified entries.
CompareCollab	Designed to find out the presence/absence of specified attributes.
ModifyCollab	Designed to add attributes, remove attributes and replace attribute values.
RenameCollab	Designed to rename an entry in the directory.
RemoveCollab	Designed to remove an entry from the directory.
SearchCollab_Persistent	Designed demonstrates the persistent search control mechanism.
SearchCollab_TimestampSearch	Designed alternative to persistent search control where this control is not supported. Uses the timestamp of entries to detect changes.
SearchCollab_SSL	Designed search entry scenario using SSL connection.
SearchCollab_StartTLS	Designed search entry scenario using Start TLS extension.

Sample Project Components

The **LDAP_SampleProject_510** sample Project includes the following components:

- File external application (inbound File eWay): **FileIn**
- File external application (outbound File eWay): **FileOut**
- Business logic implementation employs a Java-based Collaboration (eGate Service component) for processing data: **LDAP_Service**
- External LDAP system (LDAP eWay): **LDAP_System**

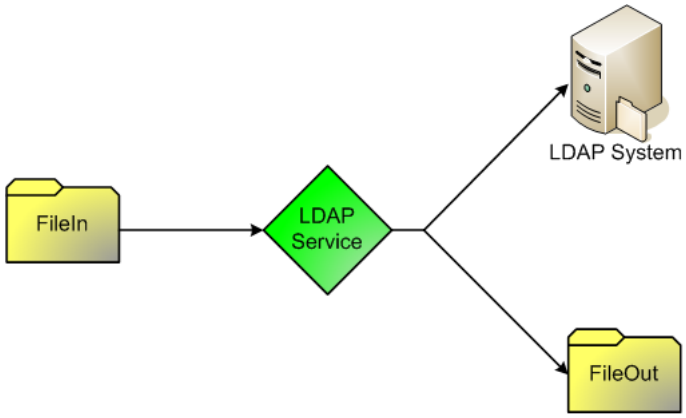
Sample Project Operation

The sample Project operates as follows:

- **FileIn**: The external file system (inbound File eWay) provides query and selection instructions to the inbound File eWay; this eWay gets a text file containing the instructions and passes them to a Java-based Collaboration Service, **LDAP_Service**.
- **LDAP_Service**: The **LDAP_Service** sends instructions to the desired LDAP system via the LDAP eWay. **LDAP_Service** also receives the information from the LDAP system, via the LDAP eWay, then sends it to a File eWay, **FileOut**.
- **LDAP_System**: The LDAP eWay handles inbound and outbound communication with this system.
- **FileOut**: The external file system (outbound File eWay) that receives the query information; another File eWay writes the received information to a text file on this system.

The following illustration shows how this scenario operates.

Figure 23 Sample Project Scenario



6.1.3 XML File Naming Conventions

The sample input **XML** files are also named according to the performed operation. Each file name ends with the two-letter code for the directory server to which the data pertains, as follows:

- **AD**: Active Directory
- **OL**: OpenLDAP
- **SO**: Sun ONE

For example, the **input_ldapmodify_replaceattr_SO.xml** file is used with the modify operation to replace an attribute: This type of operation must be used with the Sun ONE directory server.

Place these **XML** files in the desired location and set properties for the input File eWay accordingly. For more information, see **Configuring the eWay Properties** on page 94.

6.1.4 Sample Project Directory Structure

Running the sample Project requires entering the input data into the directory server using the **LDIF** files found in the **input_data** folder.

Sending data to a prescribed location requires creating the following directory structure:

Active Directory

Location: **Data\LDIF\ActiveDirectory**

Files:

- **ldap_AD.ldif** (creates top-level nodes)
- **ldap_AD_adduser.ldif** (adds users to the top-level nodes)

OpenLDAP

Location: **Data\LDIF\OpenLDAP**

Files:

- **ldap_openldap.ldif**

Sun ONE

Location: **Data\LDIF\SunOne**

Files:

- **ldap_sunone.ldif**

For complete upload procedures, refer to the appropriate system administrator's guide for the current directory server. For more information on the directory servers, see the Web sites for Active Directory, OpenLDAP, and Sun ONE.

Additional Information

The data and directory structure in an .**LDIF** file provides only the basic database skeleton. This structure does not include the data entry for the sample Project. You must provide the appropriate input data, because the result depends on the data in the LDAP database. If data that does not match the LDAP database is included in or requested by the input data file, the current Collaboration may fail.

For example, if you run a search operation without running an add operation first, this omission can cause the search operation to fail. This failure happens because the data being searched for does not exist in the LDAP database.

When you are preparing an input file for a particular action, you need to open the file and compare the data in the file with the data in the current LDAP database to make sure you are performing a valid operation, for example:

- Before doing an add operation, make sure the desired addition has not already been done.
- Before doing a search, make sure the entry in the input file exists in the database.
- Before doing a remove operation, make sure there is a matching entry in the database to be removed.

If any of these logical rules is violated, the system throws an exception.

Order of Operations: It is recommended that you run the sample Project in the following order of operations:

- Add
- Search, compare, or modify (add attribute, replace attribute, and remove attribute)
- Rename
- Remove

You can run Persistent Search and Timestamp Search at any time, but these operations are only triggered by changes in the database resulting from a particular event.

6.2 Steps Required to Run the Sample Project

The following steps are required to run the sample projects contained in the **LDAPeWayDocs.sar** file.

1. Enter the input data into the directory server using the **LDIF** files found in the **input_data** folder. For instructions on creating the required file structure, see **Sample Project Directory Structure** on page 83.
2. Import the sample Projects.
3. Build, deploy, and run the sample Projects.

   You must do the following before you can run an imported sample Project:

   - Create an Environment
   - Configure the eWays
   - Create a Deployment Profile
   - Create and start a domain
   - Deploy the Project
4. Check the output.

6.3 Importing a Sample Project

Sample eWay Projects are included as part of the installation package. To import a sample eWay Project to the Enterprise Designer do the following:

1. Extract the samples from the Java Composite Applicaton Platform Suite Installer to a local file.

   Sample files are uploaded with the eWay's documentation SAR file, and then downloaded from the Installer's Documentation tab. The **LDAP_SampleProject_510** file contains the various sample Project ZIP files.

Note: *Make sure you save all unsaved work before importing a Project.*

2 From the Enterprise Designer's Project Explorer pane, right-click the Repository and select **Import Project** from the shortcut menu. The **Import Manager** appears.

3 Browse to the directory that contains the sample Project ZIP file. Select the sample file and click **Import**.

4 Click **Close** after successfully importing the sample Project.

6.4 Building, Deploying, and Running the Sample Project

The following provides step-by-step instructions for manually creating the **LDAP_SampleProject_510** sample Project.

Steps required to create the sample project include:

6.4.1 Creating a Project

The first step is to create a new Project in the Enterprise Designer.

1 Start the Enterprise Designer.

2 From the Project Explorer tree, right-click the Repository and select **New Project**. A new Project (**Project1**) appears on the Project Explorer tree.

3 Click twice on **Project1** and rename the Project (for this sample, **LDAP_SampleProject_510**).

6.4.2 Creating the OTDs

The sample Project requires an OTD to interact with the LDAP eWay.

Steps required to create an LDAP OTD:

1. Right-click your new Project in the Enterprise Designer's Project Explorer, and select **New** > **Object Type Definition**.

   The New Object Type Definition Wizard window appears.
2. Select **DTD** from the list of OTD Wizards and click **Next**.
3. Browse to the location of the DTD files included in the **LDAP_SampleProject_510** file and select the **ldapsearch.dtd** file (located in the **Input_Data** > **SearchEntry** folders).
4. Click **Next**. The **ldapsearch_ldapsearch** becomes a highlighted document element.
5. Click **Next**. The Select OTD Options window appears.
6. Click **Finish** to create the OTD.

6.4.3 Creating the Collaboration Definitions (Java)

The next step is to create Collaboration Definitions (Java) or JCDs using the **Collaboration Definition Wizard (Java)**. Once you create a Collaboration Definition, you can write the Business Rules using the Collaboration Editor.

Steps required to create the Collaboration:

1. From the Project Explorer, right-click the sample Project and select **New** > **Collaboration Definition (Java)** from the shortcut menu. The **Collaboration Definition Wizard (Java)** appears.
2. Enter a Collaboration Definition name (for this sample **SearchCollab**) and click **Next**.
3. For Step 2 of the wizard, from the Web Services Interfaces selection window, double-click **Sun SeeBeyond** > **eWays** > **File** > **FileClient** > **receive**. The File Name field now displays **receive.** Click **Next**.
4. For Step 3 of the wizard, from the Select OTDs selection window, double-click **LDAP_SampleProject_510** > **otdALL** > **ldapsearch_ldapsearch**. The **ldapsearch_ldapsearch** OTD is added to the Selected OTDs field.
5. Click the **Up One Level** button twice to return to the Repository. Double-click **Sun SeeBeyond** > **eWays** > **File** > **FileClient**. The **Selected OTDs** field now lists the **FileClient_1** OTD.
6. Click the **Up One Level** button and select LDAP > LDAPClient. The **Selected OTDs** field now lists the **LDAPClient_1** OTD.
7. Click **Finish**. The Collaboration Editor with the new **SearchCollab** Collaboration appears in the right pane of the Enterprise Designer.

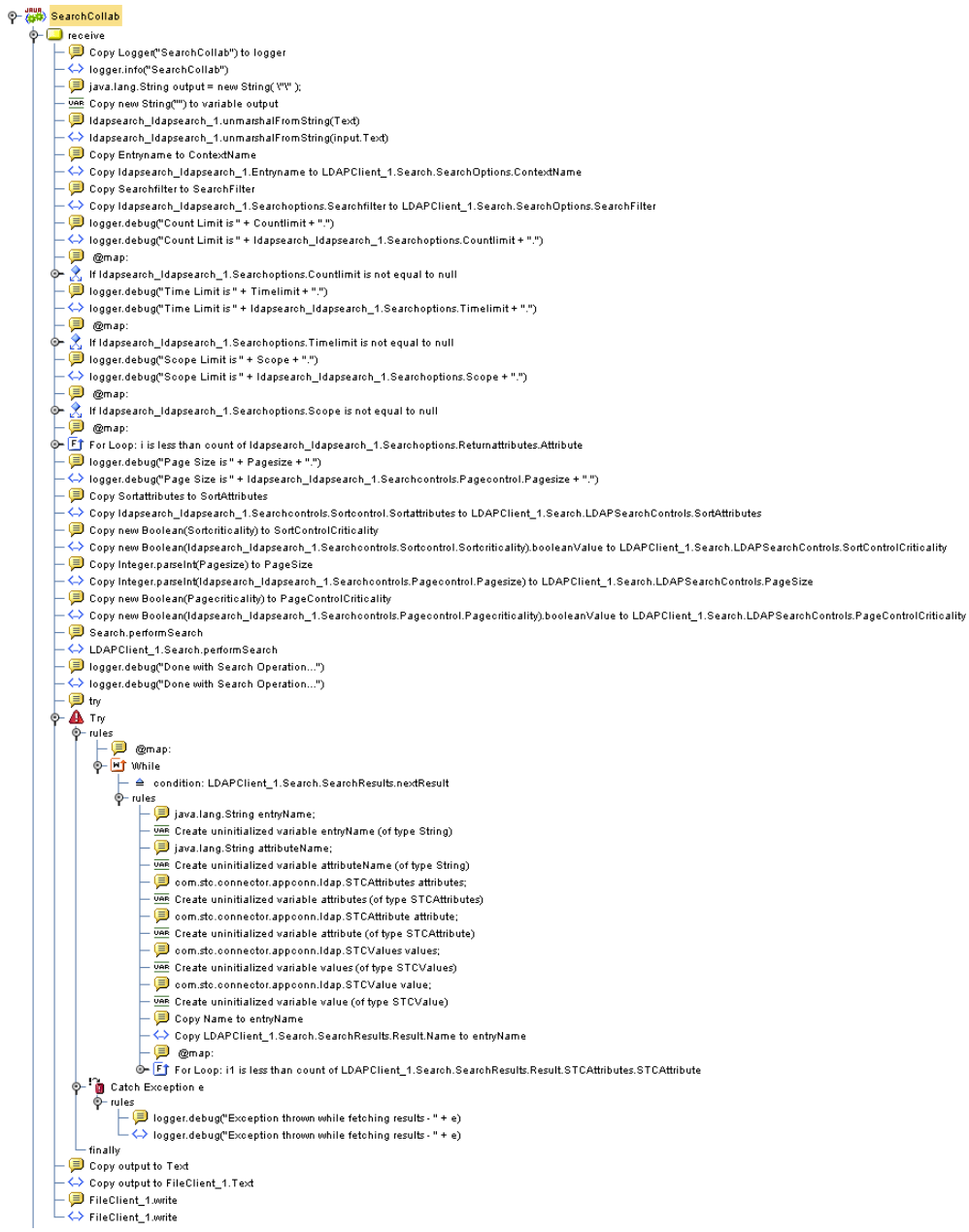
6.4.4 Create the Collaboration Business Rules

The next step in the sample is to create the Business Rules of the Collaboration using the Collaboration Editor.

In this sample Project, the Collaboration receives query and selection instructions from the external file system (FileIn). This information is then passed to the LDAP system via the LDAP eWay. The external file system (FileOut) then writes the received information to a text-based file on the system.

The **SearchCollab** Collaboration contains the Business Rules displayed in Figure 24.

Figure 24 SearchCollab Business Rules



Sample code from the SearchCollab Includes:

```
package LDAP_SampleProject_510;

public class SearchCollab
{
```

```
    public com.stc.codegen.logger.Logger logger;
    public com.stc.codegen.alerter.Alerter alerter;
    public com.stc.codegen.util.CollaborationContext collabContext;
    public com.stc.codegen.util.TypeConverter typeConverter;


    public void receive(
    com.stc.connector.appconn.file.FileTextMessage input,
    com.stc.connector.appconn.file.FileApplication FileClient_1,
    com.stc.connector.appconn.ldap.LDAPClientApplication LDAPClient_1,
    ldapsearch.Ldapsearch ldapsearch_ldapsearch_1 )
    throws Throwable
    {
// @map:Copy Logger("SearchCollab") to logger
    logger.info( "SearchCollab" );
// @map:java.lang.String output = new String( \"\" );
    String output = new String( "" );
// @map:ldapsearch_ldapsearch_1.unmarshalFromString(Text)
    ldapsearch_ldapsearch_1.unmarshalFromString( input.getText() );
// @map:Copy Entryname to ContextName
    LDAPClient_1.getSearch().getSearchOptions().setContextName(
    ldapsearch_ldapsearch_1.getEntryname() );
// @map:Copy Searchfilter to SearchFilter
    LDAPClient_1.getSearch().getSearchOptions().setSearchFilter(
    ldapsearch_ldapsearch_1.getSearchoptions().getSearchfilter() );
// @map:logger.debug("Count Limit is " + Countlimit + ".")
    logger.debug( "Count Limit is " +
    ldapsearch_ldapsearch_1.getSearchoptions().getCountlimit() + "."
    );
// @map:
    if (ldapsearch_ldapsearch_1.getSearchoptions().getCountlimit() !=
    null) {
// @map:Copy Integer.parseInt(Countlimit) to CountLimit
    LDAPClient_1.getSearch().getSearchOptions().setCountLimit(
    Integer.parseInt(
    ldapsearch_ldapsearch_1.getSearchoptions().getCountlimit() ) );
}

// @map:logger.debug("Time Limit is " + Timelimit + ".")
    logger.debug( "Time Limit is " +
    ldapsearch_ldapsearch_1.getSearchoptions().getTimelimit() + "." );
// @map:
    if (ldapsearch_ldapsearch_1.getSearchoptions().getTimelimit() !=
    null) {
// @map:Copy Integer.parseInt(Timelimit) to TimeLimit
    LDAPClient_1.getSearch().getSearchOptions().setTimeLimit(
    Integer.parseInt(
    ldapsearch_ldapsearch_1.getSearchoptions().getTimelimit() ) );
    }
// @map:logger.debug("Scope Limit is " + Scope + ".")
    logger.debug( "Scope Limit is " +
    ldapsearch_ldapsearch_1.getSearchoptions().getScope() + "." );
// @map:
    if (ldapsearch_ldapsearch_1.getSearchoptions().getScope() != null)
    {
// @map:Copy Integer.parseInt(Scope) to SearchScope
    LDAPClient_1.getSearch().getSearchOptions().setSearchScope(
    Integer.parseInt(
    ldapsearch_ldapsearch_1.getSearchoptions().getScope() ) );
    }
// @map:
    for (int i = 0; i <
    ldapsearch_ldapsearch_1.getSearchoptions().getReturnattributes().c
    ountAttribute(); i++) {
```

```
// @map:AttributesSelection.addAttribute(Attribute(i))
   LDAPClient_1.getSearch().getSearchOptions().getAttributesSelection
   ().addAttribute(
   ldapsearch_ldapsearch_1.getSearchoptions().getReturnattributes().g
   etAttribute( i ) );
   }
// @map:logger.debug("Page Size is " + Pagesize + ".")
   logger.debug( "Page Size is " +
   ldapsearch_ldapsearch_1.getSearchcontrols().getPagecontrol().getPa
   gesize() + "." );
// @map:Copy Sortattributes to SortAttributes
   LDAPClient_1.getSearch().getLDAPSearchControls().setSortAttributes
   (
   ldapsearch_ldapsearch_1.getSearchcontrols().getSortcontrol().getSo
   rtattributes() );
// @map:Copy new Boolean(Sortcriticality) to SortControlCriticality
   LDAPClient_1.getSearch().getLDAPSearchControls().setSortControlCri
   ticality( (new Boolean(
   ldapsearch_ldapsearch_1.getSearchcontrols().getSortcontrol().getSo
   rtcriticality() )).booleanValue() );
// @map:Copy Integer.parseInt(Pagesize) to PageSize
   LDAPClient_1.getSearch().getLDAPSearchControls().setPageSize(
   Integer.parseInt(
   ldapsearch_ldapsearch_1.getSearchcontrols().getPagecontrol().getPa
   gesize() ) );
// @map:Copy new Boolean(Pagecriticality) to PageControlCriticality
   LDAPClient_1.getSearch().getLDAPSearchControls().setPageControlCri
   ticality( (new Boolean(
   ldapsearch_ldapsearch_1.getSearchcontrols().getPagecontrol().getPa
   gecriticality() )).booleanValue() );
// @map:Search.performSearch
   LDAPClient_1.getSearch().performSearch();
// @map:logger.debug("Done with Search Operation...")
   logger.debug( "Done with Search Operation..." );
// @map:try
   try {
// @map:
   while (LDAPClient_1.getSearch().getSearchResults().nextResult()) {
// @map:java.lang.String entryName;
   String entryName;
// @map:java.lang.String attributeName;
   String attributeName;
// @map:com.stc.connector.appconn.ldap.STCAttributes attributes;
   com.stc.connector.appconn.ldap.STCAttributes attributes;
// @map:com.stc.connector.appconn.ldap.STCAttribute attribute;
   com.stc.connector.appconn.ldap.STCAttribute attribute;
// @map:com.stc.connector.appconn.ldap.STCValues values;
   com.stc.connector.appconn.ldap.STCValues values;
// @map:com.stc.connector.appconn.ldap.STCValue value;
   com.stc.connector.appconn.ldap.STCValue value;
// @map:Copy Name to entryName
   entryName =
   LDAPClient_1.getSearch().getSearchResults().getResult().getName();
// @map:
   for (int i1 = 0; i1 <
   LDAPClient_1.getSearch().getSearchResults().getResult().getSTCAttr
   ibutes().countSTCAttribute(); i1 = i1 + 1) {
// @map:
   for (int i2 = 0; i2 <
   LDAPClient_1.getSearch().getSearchResults().getResult().getSTCAttr
   ibutes().getSTCAttribute( i1 ).getSTCValues().countSTCValue(); i2
   = i2 + 1) {
// @map:Copy Name to attributeName
```

```
   attributeName =
   LDAPClient_1.getSearch().getSearchResults().getResult().getSTCAttr
   ibutes().getSTCAttribute( i1 ).getName();
// @map:Copy STCValue(i2) to value
   value =
   LDAPClient_1.getSearch().getSearchResults().getResult().getSTCAttr
   ibutes().getSTCAttribute( i1 ).getSTCValues().getSTCValue( i2 );
// @map:Copy "Entry Name is - " + entryName + "\n" to output
   output += "Entry Name is - " + entryName + "\n";
// @map:Copy "Attribute Name is - " + attributeName + "\n" to output
   output += "Attribute Name is - " + attributeName + "\n";
// @map:Copy StringValue to output
   output +=
   LDAPClient_1.getSearch().getSearchResults().getResult().getSTCAttr
   ibutes().getSTCAttribute( i1 ).getSTCValues().getSTCValue( i2
   ).getStringValue();
// @map:Copy "\n" to output
   output += "\n";
   }
   }
   }
   } catch ( Exception e ) {
// @map:logger.debug("Exception thrown while fetching results - " +
e)
   logger.debug( "Exception thrown while fetching results - " + e );
   }
// @map:Copy output to Text
   FileClient_1.setText( output );
// @map:FileClient_1.write
   FileClient_1.write();
   }

   }
```

6.4.5 Creating a Connectivity Map

The Connectivity Map provides a canvas for assembling and configuring a Project's components.

Steps required to create a new Connectivity Map:

1. From the Project Explorer tree, right-click the new **LDAP_SampleProject_510** Project and select **New > Connectivity Map** from the shortcut menu.
2. The New Connectivity Map appears and a node for the Connectivity Map is added under the Project, on the Project Explorer tree labeled **CMap1**. Rename this project to be **CMap**.

Populating the Connectivity Map

Add the Project components to the Connectivity Map by dragging the icons from the toolbar to the canvas.

Each Connectivity Map in the **LDAP_SampleProject_510** sample Project requires the following components:

- File External Application (x2)
- LDAP External Application

- Service

Any eWay added to the Connectivity Map is associated with an External System. To establish a connection to LDAP, first select LDAP as an External System to use in your Connectivity Map.

Steps required to select a LDAP External System:

1. Click the **External Application** icon on the Connectivity Map toolbar.
2. Select the external systems necessary to create your Project (for this sample, **LDAP** and **File**). Icons representing the selected external systems are added to the Connectivity Map toolbar.
3. Rename the following components and then save changes to the Repository:
   - File1 to FileIn
   - File2 to FileOut
   - LDAP1 to LDAP_System

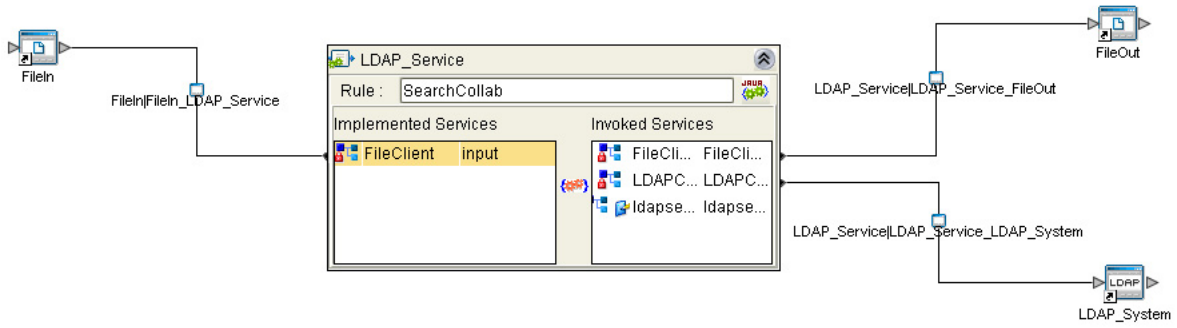
6.4.6 Binding the eWay Components

The final step in creating a Connectivity Map is binding the eWay components together.

Steps required to manually bind eWay components together:

1. Double-click a Connectivity Map—in this example **CMap**—in the Project Explorer tree. The **CMap** Connectivity Map appears in the Enterprise Designers canvas.
2. Drag and drop the **SearchCollab** Collaboration from the Project Explorer to **LDAP_Service**. The Service icon "gears" change from red to green.
3. Double-click **LDAP_Service**. The **LDAP_Service** Binding dialog box appears.
4. Map the input **FileClient** (under Implemented Services) to the **FileIn** (File) External Application. To do this, drag the **FileClient** OTD in the **LDAP_Service** Binding dialog box to the **FileIn** External Application in the Connectivity Map. A link is now visible between **FileIn** and **FileClient**.
5. From the same Binding dialog box, map **FileClient** (under Invoked Services) to the **FileOut** External Application. A link is now visible between **FileOut** and **FileClient**.
6. From the same Binding dialog box, map **LDAPClient** to the **LDAP_System** External Application, as seen in Figure 25.

Figure 25 Connectivity Map - Associating (Binding) the Project's Components



7 Minimize the **LDAP_Service** Binding dialog box by clicking the chevrons in the upper-right corner.

8 Save your current changes to the Repository.

6.4.7 Creating an Environment

Environments include the external systems, Logical Hosts, integration servers and message servers used by a Project and contain the configuration information for these components. Environments are created using the Enterprise Designer's Environment Editor.

Steps required to create an Environment:

1 From the Enterprise Designer's Enterprise Explorer, click the **Environment Explorer** tab.

2 Right-click the Repository and select **New Environment**. A new Environment is added to the Environment Explorer tree.

3 Rename the new Environment to **envLDAPProj**.

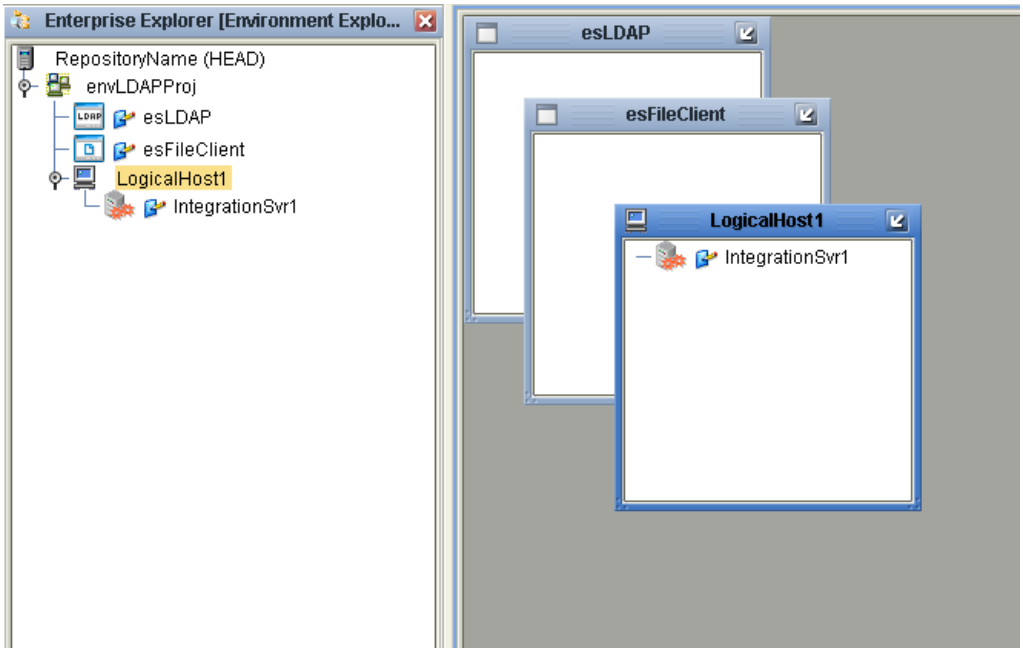
4 Right-click **envLDAPProj** and select **LDAP External System**. Name the External System **esLDAP.** Click **OK**. **esLDAP** is added to the Environment Editor.

5 Right-click **envLDAPProj** and select **File External System**. Name the External System **esFileClient**. Click **OK**. **esFileClient** is added to the Environment Editor.

6 Right-click **envLDAPProj** and select **Logical Host**. The **LogicalHost1** box is added to the Environment and **LogicalHost1** is added to the Environment Editor tree.

7 Right-click **LogicalHost1** and select **Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under **LogicalHost1**.

Figure 26 Environment Editor - envLDAPProj



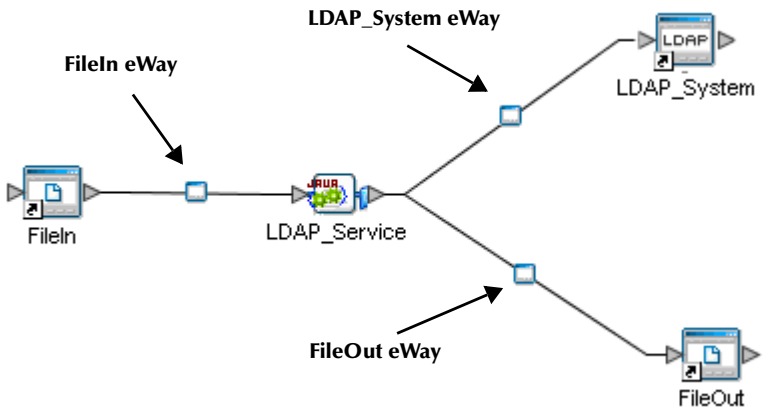
8 Save your current changes to the Repository.

6.4.8 Configuring the eWays

eWays facilitate communication and movement of data between the external applications and the eGate system. The Connectivity Map in the sample Project uses three eWays, represented as nodes between the External Applications and the Business Process, as seen in Figure 27.

You must configure eWay properties in both the Connectivity Map and the Environment Explorer.

Figure 27 eWays in the cmDelete Connectivity Map



Configuring the eWay Properties

Steps required to configure the eWay properties:

1. Double-click the **FileIn eWay** and modify the following property for your system:

   Input File Name: **<filename>.fin**

2. Double-click the **FileOut eWay** and modify the following property for your system:

   Output File Name: **LDAP_output%d.dat**

3. Double-click the LDAP_System eWay and modify the following properties under the Connection section for your system:

   Provider URL: The LDAP server and port number you are using.

   Authentication: Set this if your server does not accept anonymous login. In this case you may also need to set the Principle and Credentials fields.

Steps required to configure the Environment Explorer properties:

1. From the **Environment Explorer** tree, right-click the File External System (**esFileClient** in this sample), and select **Properties**. The Properties Editor opens to the File eWay Environment configuration.
2. Modify the Parameter settings as required for your Environment, and click **OK**.

Note: *To run this sample Project, you do not need to change the default Environment Configuration properties of the LDAP External.*

Note: *See* **"Setting LDAP eWay Properties"** *for additional configuration properties of the LDAP eWay.*

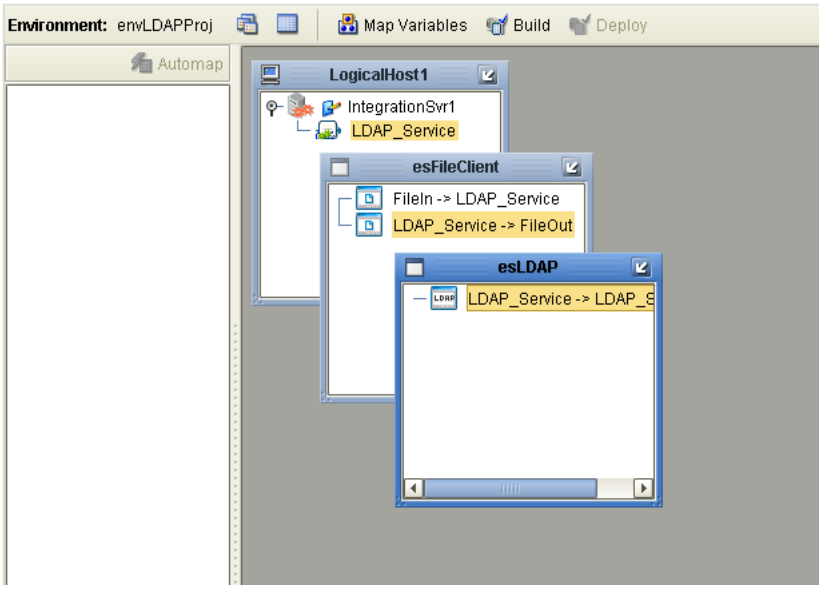
6.4.9 Creating the Deployment Profile

A Deployment Profile is used to assign services and message destinations to the integration server and message server. Deployment profiles are created using the Deployment Editor.

Steps required to create the Deployment Profile:

1. From the Enterprise Explorer's Project Explorer, right-click the **LDAP_SampleProject_510** Project and select **New** > **Deployment Profile**.
2. Enter a name for the Deployment Profile (for this sample **dpLDAP**). Select **envLDAPProj** as the Environment and click **OK**.
3. From the Deployment Editor toolbar, click the **Automap** icon. The Project's components are automatically mapped to their system windows, as seen in.

Figure 28 Deployment Profile



6.4.10 Creating and Starting the Domain

A domain is an instance of a Logical Host. After the domain is created, the Project is built and then deployed.

Note: *You are only required to create a domain once when you install the Java Composite Application Platform Suite.*

Steps required to create and start the domain:

1. Navigate to your **<JavaCAPS51>\logicalhost** directory (where <JavaCAPS51> is the location of your Java Composite Application Platform Suite installation.
2. Double-click the **domainmgr.bat** file. The **Domain Manager** appears.
3. If you have already created a domain, select your domain in the Domain Manager and click the **Start an Existing Domain** button. Once your domain is started, a green check mark indicates that the domain is running.
4. If there are no existing domains, a dialog box indicates that you can create a domain now. Click **Yes**. The **Create Domain** dialog box appears.
5. Make any necessary changes to the **Create Domain** dialog box and click **Create**. The new domain is added to the Domain Manager. Select the domain and click the **Start an Existing Domain** button. Once your domain is started, a green check mark indicates that the domain is running.

For more information about creating and managing domains see the *eGate Integrator System Administration Guide*.

6.4.11 Building and Deploying the Project

The Build process compiles and validates the Project's Java files and creates the Project EAR file.

Build the Project

1. From the Deployment Editor toolbar, click the **Build** icon.
2. If there are any validation errors, a **Validation Errors** pane will appear at the bottom of the Deployment Editor and displays information regarding the errors. Make any necessary corrections and click **Build** again.
3. After the Build has succeeded you are ready to deploy your Project.

Deploy the Project

1. From the Deployment Editor toolbar, click the **Deploy** icon. Click **Yes** when the **Deploy** prompt appears.
2. A message appears when the project is successfully deployed. You can now test your sample.

Note: *There are several ways to deploy a project, for additional information, see the Sun SeeBeyond eGate™ Integrator System Administration Guide.*

6.4.12 Running the Sample

Additional steps are required to run the deployed sample Project.

Steps required to run the sample Project:

1. Rename one of the trigger files included in the sample Project from **<filename>.xml** to **<filename>.fin** to run the corresponding operation.

   The File eWay polls the directory every five seconds for the input file name (as defined in the Inbound File eWay Properties window). The JCD then transforms the data, and the File eWay sends the output to an Output file name (as defined in the outbound File eWay Properties window).

Note: *The type of XML file you choose depends on the type of directory server being used. For details, see* **XML File Naming Conventions** *on page 82.*

2. Verify the output data by viewing the sample output files. See for more details on the types of output files used in this sample Project. The output files may change depending on the number of times you execute the sample Project, the input file, and also the content of your database table.

Index

P

R

S

T

V

X