SeeBeyond ICAN Suite

LDAP eWay Intelligent Adapter User's Guide

Release 5.0

SEEBEYOND®

Version 20040423122126.

Contents

Chapter 1

Introducing the LDAP eWay

This guide explains how to use, set properties for, and operate the SeeBeyond LDAP eWay Intelligent Adapter (LDAP eWay).

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

Chapter Topics

1.1 Overview

This section provides a general overview of the LDAP eWay and its operation with eGate Integrator and the Lightweight Directory Access Protocol (LDAP) system.

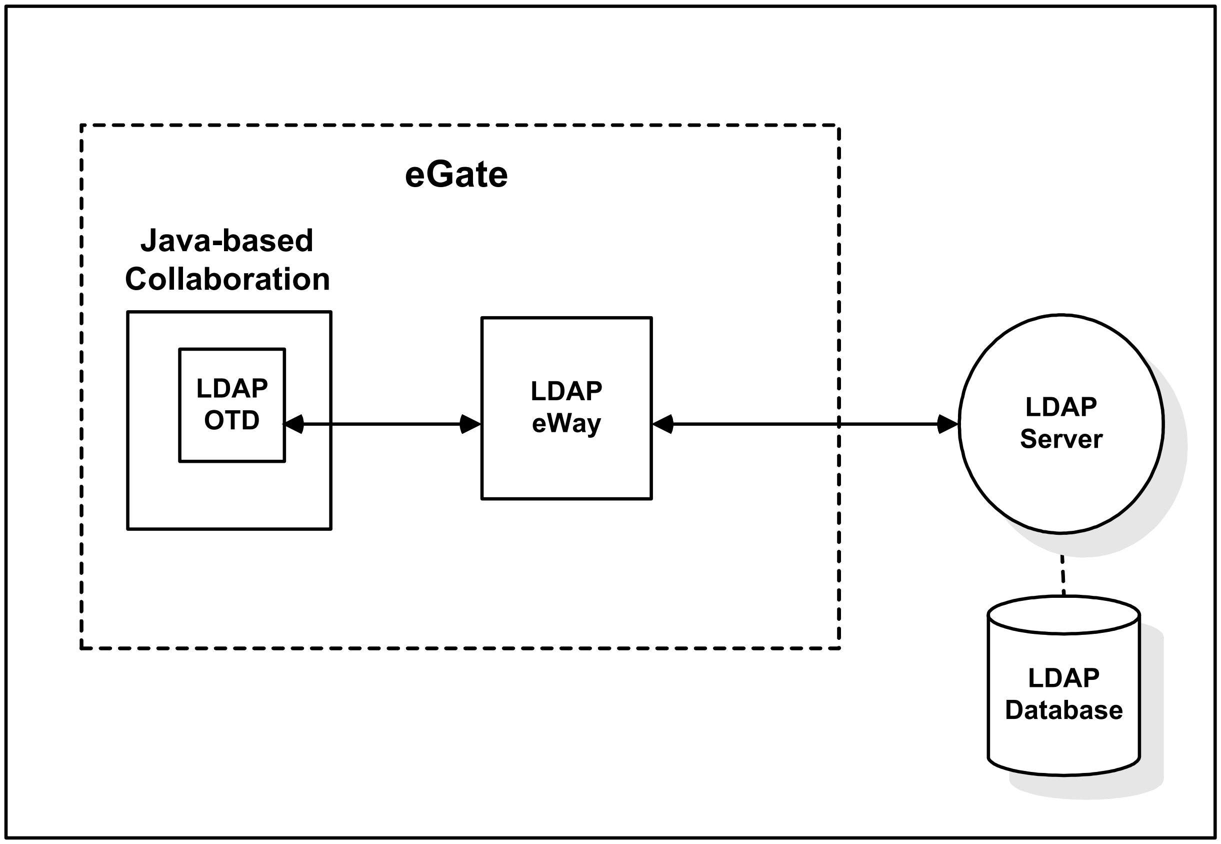
1.1.1 eWay General Operation

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 1 on page 7** for a general diagram of the architecture of the LDAP eWay.

Figure 1 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.

For another example, you can use the LDAP eWay, in conjunction with eGate, to make a company's employee directory available on an intranet Web site. With the click of a button in a Web browser, a user with appropriate permissions can then access these employee records as desired.

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.

1.1.2 Java Naming and Directory Interface

The LDAP eWay uses Sun'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 Provided Information

This section describes the general information provided by this user's guide, as well as additional information features provided, the sample Project and **Javadoc**.

1.2.1 Setting eWay Properties

The properties for the LDAP eWay allow you to set necessary parameters of operation using the eWay's **Properties** sheet in the Enterprise Designer. See **Chapter 3** for details on how to set the eWay's properties.

1.2.2 Using the LDAP OTD

For an introduction to LDAP and an explanation of how to access LDAP functions using the eWay's LDAP OTD, see **Chapter 4**.

1.2.3 Sample Project

A sample Project for the LDAP eWay is included on the installation CD-ROM. This sample demonstrates how to implement basic LDAP scenarios in eGate. See **Chapter 2** for information on how to access the sample Project. See **Chapter 5** for information on the Project.

1.2.4 Javadoc

A **Javadoc** for the LDAP eWay is included on the installation CD-ROM. This document explains the Java methods available with the eWay. See **Chapter 2** for information on how to access the **Javadoc**. See **Chapter 6** for information on the **Javadoc**.

1.3 Supported Operating Systems

The LDAP eWay is available on the following operating systems:

- Windows XP, Windows 2000, and Windows Server 2003
- HP Tru64 V5.1A
- HP-UX 11.0 and 11i (PA-RISC)
- IBM AIX 5.1L and 5.2
- Red Hat Linux 8 (Intel) and Advanced Server 2.1 (Intel)
- Sun Solaris 8 and 9

1.4 System Requirements

To use the LDAP eWay, you need:

- eGate Logical Host.
- TCP/IP network connection.

Logical Host Requirements

The eWay must have its properties set and be administered using the eGate Enterprise Designer. For complete information on the Enterprise Designer system requirements, see the *SeeBeyond ICAN Suite Installation Guide*.

1.5 External System Requirements

You need an LDAP server supporting LDAP version 3.0 on any of the following LDAP server types:

- Windows 2000 Server Active Directory
- Sun ONE Directory Server version 5.2
- OpenLDAP version 2.1

To enable the eWay to communicate with the LDAP system, you need:

- Host on which the LDAP server is running.
- Port location on which the LDAP server is listening.
- Authentication information.
- Understanding of LDAP directory structure being used.

Chapter 2

Installing the LDAP eWay

This chapter explains how to install the LDAP eWay.

Chapter Topics

2.1 Installation Procedures

During the ICAN Suite installation process, the Enterprise Manager, a Web-based application, is used to select and upload eWay and add-on files (**.sar** files) from the ICAN installation CD-ROM to the Repository.

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

Before installing the eWay

Open and review the **Readme.txt** file (located in the root directory of the ICAN installation's Repository CD-ROM) for the latest information, before installing the eWay.

Installing the LDAP eWay on an eGate-supported system

The LDAP eWay can be installed during the installation of eGate. The eGate installation process includes the following operations:

- Installing the eGate Repository
- Uploading products to the Repository
- Downloading the components (including the eGate Enterprise Designer and the Logical Host)
- Viewing the product information home pages

Follow the instructions for installing the ICAN Suite found in the *SeeBeyond ICAN Suite Installation Guide*, and include the following steps:

1. After the eGate or eInsight core products are uploaded to the Repository using the Enterprise Manager, select and upload the **FileeWay.sar**. The File eWay is used by the eWay's Project sample. You must upload the File eWay (**FileeWay.sar**.) before uploading the LDAP eWay (**LDAPeWay.sar**).
2. After the File eWay is uploaded, upload **LDAPeWay.sar** to install the LDAP eWay.
3. Next, upload the **LDAPeWayDocs.sar**. This file contains the eWay user's guide, **Javadoc**, and sample Project files.

   To obtain these files, follow the instructions provided by the user interface.
4. If needed, continue installing eGate as instructed in the *SeeBeyond ICAN Suite Installation Guide.*

2.2 After Installation

Once the eWay is installed and configured, it must then be incorporated into a Project before it can perform its intended functions. See the *eGate Integrator User's Guide* for more information on incorporating the eWay into an eGate Project.

Chapter 3

Setting LDAP eWay Properties

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

Chapter Topics

3.1 LDAP eWay Properties Sheet

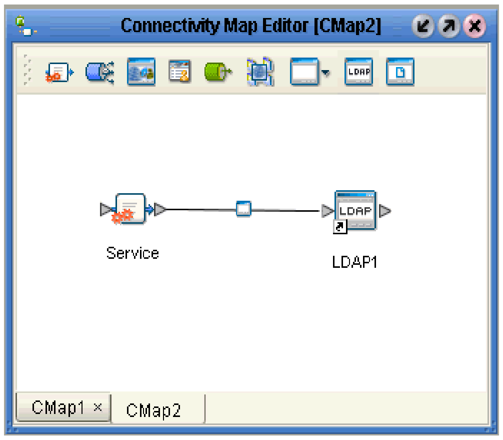
When you install the LDAP eWay, a default properties template for the eWay is also installed. The template's default properties are accessible using the eGate Enterprise Designer. These default settings apply to all LDAP eWays you use within your current Project.

You can set properties for the eWay using the Enterprise Designer's eWay **Properties** sheet. This section describes general procedures on how to change these default properties for the eWay. For details on these steps, see the *eGate Integrator User's Guide*.

To set properties for the LDAP eWay on the Connectivity Map

1. From the eGate Enterprise Designer's **Project Explorer** create at least one Connectivity Map.
2. Create the desired external systems for your one or more Connectivity Maps.
3. Select the external application whose default eWay properties you want to change by double-clicking the **eWay** icon. This icon is located on the link between an **External Application** icon and a **Service** icon on the Connectivity Map canvas. See **Figure 2 on page 13**.

Figure 2 eWay Icon



The eWay **Properties** sheet appears. **Figure 3 on page 14** shows the eWay's default properties available via the **Project Explorer** and Connectivity Map. You can use this window to modify the current eWay's properties settings.

4 Click **OK** then **Save All** to save your changes.

To use the eWay Properties sheet

- The eWay's default properties are automatically provided. You can change them, as desired.
- Clicking the **Configuration** folder in the left pane displays the properties group subfolder in the right pane. Click any subfolder to display the eWay's editable properties.
- Many of the entries allow you to enter text. Click the desired text box, then click the ellipsis (**...**) that appears, to open a dialog box for this purpose.

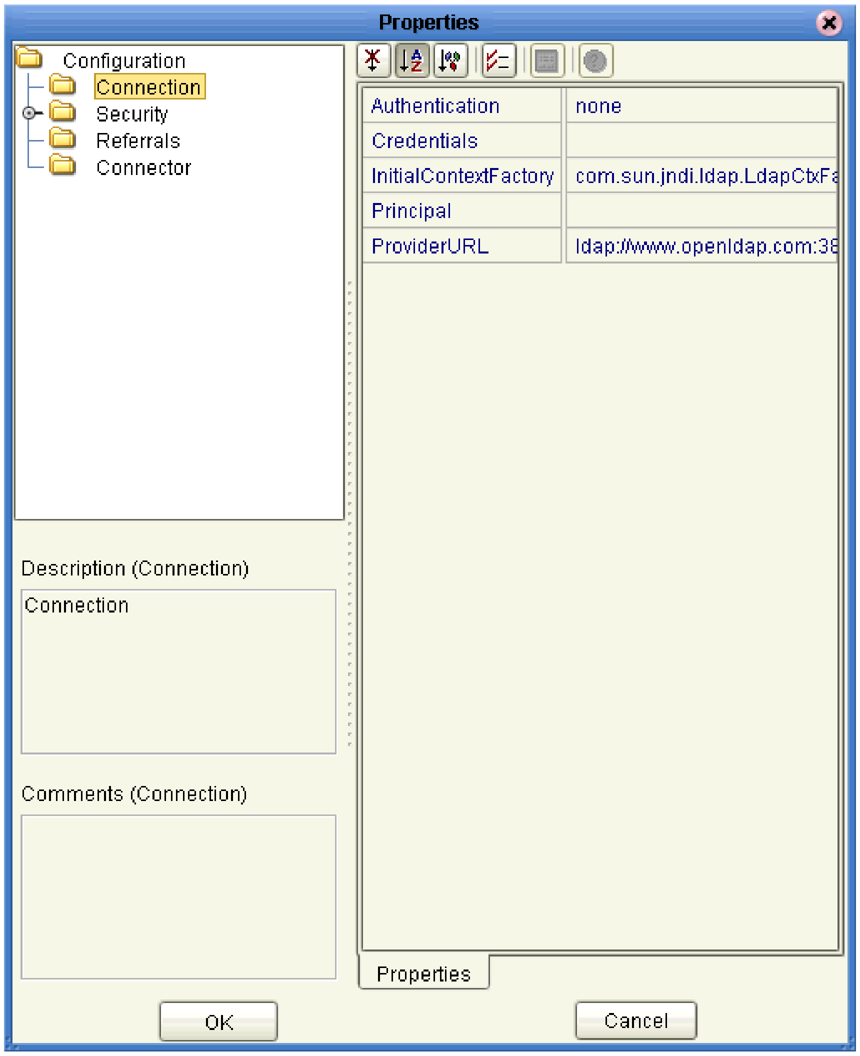
Note: *Even if you do not change the eWay's properties, you must open each* ***Properties*** *sheet for each eWay and click* ***OK*** *to activate the eWay.*

The remainder of this chapter explains the eWay's properties in detail.

3.2 Setting eWay Properties on the Connectivity Map

This section explains in detail the eWay's editable properties accessible on the Connectivity Map via the eGate Enterprise Designer's **Project Explorer**. You can set these properties using the eWay **Properties** sheet. See Figure 3.

Figure 3 eWay Properties Sheet: Settings on the Connectivity Map



Connection

Authentication	none
Credentials	
InitialContextFactory	com.sun.jndi.ldap.LdapCtxFa
Principal	
ProviderURL	ldap://www.openldap.com:38

The eWay's properties settings define the properties used to interact with the external system, under the **Configuration** folder. These settings are:

- **"Connection" on page 15**
- **"Security/SSL" on page 16**
- **"Referrals" on page 20**
- **"Connector" on page 23**

3.2.1 Connection

These properties allow you to define the connection to the LDAP system.

Authentication

Description

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**).

Required Values

Select **none** or **simple**; the default is **none**.

Credentials

Description

Allows you to enter the credentials needed when using an authentication mechanism other than anonymous log-in (authentication = **none**).

Required Values

The appropriate credentials, in the form of a valid password.

InitialContextFactory

Description

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.

Required Values

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

Description

Allows you to specify the principal needed when using an authentication mechanism other than anonymous log-in (authentication = **none**).

Required Values

The fully qualified Distinguished Name (DN) of the user, for example:

CN=Administrator,CN=Users,DC=stc,dc=com

ProviderURL

Description

Allows you to specify the URL of the LDAP Server.

Required Values

A valid URL with the protocol as **ldap**; the default is **ldap://www.openldap.com:389**.

3.2.2 Security/SSL

The properties explained in this section are used to set the basic security features for SSL.

JSSE Provider Class

Description

Specifies the fully qualified name of the JSSE provider class. For more information, see the Sun Microsystems Java site at:

http://java.sun.com/

Required Values

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

Description

Specifies the default KeyStore file. The keystore is used for key/certificate management when establishing SSL connections.

Required Values

A valid package location; there is no default.

KeyStore password

Description

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.

KeyStore type

Description

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.

KeyStore username

Description

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.*

SSL Connection Type

Description

Allows you to specify the type of SSL connection to be used.

Required Values

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).*

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 49**.

SSL Protocol

Description

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.

Required Values

Select **TLS**, **TLSv1**, **SSLv3**, **SSLv2**, **or SSL**.

TrustStore

Description

Specifies the default TrustStore. The TrustStore is used for CA certificate management when establishing SSL connections.

Required Values

A valid **TrustStore** name; there is no default.

TrustStore password

Description

Allows you to specify the default TrustStore password. The password is for accessing the TrustStore used for CA certificate management when establishing SSL connections.

Required Values

A valid **TrustStore** password; there is no default.

TrustStore type

Description

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.

Required Values

A valid **TrustStore** type.

Verify hostname

Description

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.

Required Values

True or **False**; the default is **False**.

Additional information

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**.*

X509 Algorithm Name

Description

Specifies the X509 algorithm name to use for the trust and key manager factories.

Required Values

The name of a valid X509 algorithm; the default is **SunX509**. If you are running the Integration Server on AIX, specify **IbmX509**.

3.2.3 Referrals

These properties allow you to enter LDAP referral information.

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 1 on page 21** can arise depending on the properties provided for the referrals and the behavior of the eWay, as explained for each of these scenarios.

Table 1 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. Extract the files to be used for the RCF utility from the following locations:
   - **<edesigner_home>\usrdir\modules\ext\ldapadapter\gnu.getopt.jar**
   - Extract the **.jar** file named **stcldap13.jar** or **stcldap14.jar** from **<edesigner_home>\usrdir\modules\ext\ldapadapter\ldapadapter.jar**
2. Place all these files (**gnu.getopt.jar**, plus **stcldap13.jar** or **stcldap14.jar**) under a folder and run the utility from this folder as follows:

   ```
   <edesigner_home>\jdk\bin\java –cp ./gnu.getopt.jar;./stcldap13.jar
   com.stc.connector.ldapadapter.utils.RCFUtil
   ```
3. Enter the following parameters on the command line:

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

   This action requests a user name and password.
4. 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.

   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. Request the user name and password for accessing the file by entering the following information:

   ```
   <edesigner_home>\jdk\bin\java -cp ./gnu.getopt.jar;./stcldap13.jar
   com.stc.connector.ldapadapter.utils.RCFUtil --add -- samplercf.txt
   ```
2. 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.

To view the contents of the credentials file

1. Request the user name and password for accessing the file by entering the following information:

   ```
   <edesigner_home>\jdk\bin\java -cp ./gnu.getopt.jar;./stcldap13.jar
   com.stc.connector.ldapadapter.utils.RCFUtil --list -- samplercf.txt
   ```
2. Provide the user name and password given for creating the file previously.

   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=
   ```

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

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

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

The following example shows the content of a credentials file 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.
#
#Wed Mar 31 16:30:30 IST 2004
#Password for accessing this file - encrypted.
password=riEp0a9FFO0\=
#New credentials entry that was created.
localhost.stc.com\:389=cn\=Manager,dc\=stc,dc\=com;l/ZRt1cfNKc\=
#Contains host:port=principal;encrypted credentials.
#Username for accessing this file.
username=localhost
```

Credentials File

Description

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.

Required Values

A valid file and path name available to eGate.

Follow

Description

Allows you to select whether referrals returned by an LDAP server must be followed.

Required Values

Select **yes** or **no**; the default is **yes**. Enter the desired value as follows:

- **Yes**: Follow referrals.
- **No**: Referrals are not followed.

See **Table 1 on page 21** for details.

3.2.4 Connector

These properties allow you to specify the server connector class and type.

Note: *Make sure when you connect to the server that you use the fully qualified DNS name of the server.*

Connector Class

Description

Allows you to enter the Java class name of the server connector.

Required Values

A valid Java class name; the default is:

com.stc.connector.ldapadapter.LDAPEwayConnection

It is recommended that you do not change this value.

Connector Type

Description

Allows you to enter the type of the server connector.

Required Values

A valid connector type; the default is **LDAP Connector**. It is recommended that you do not change this value.

Chapter 4

Using the LDAP OTD

This chapter first provides an overview of the LDAP protocol. Then, it explains the eGate 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.

Chapter Topics

4.1 Introduction to LDAP and OTD Features

LDAP is a directory service protocol that allows you to control access to applications on your network. 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.

4.1.1 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@seebeyond.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@seebeyond.com** and **jdoe@stc.com**. In the previous example, the mail attribute contains two mail values.

4.1.2 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.

4.1.3 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.

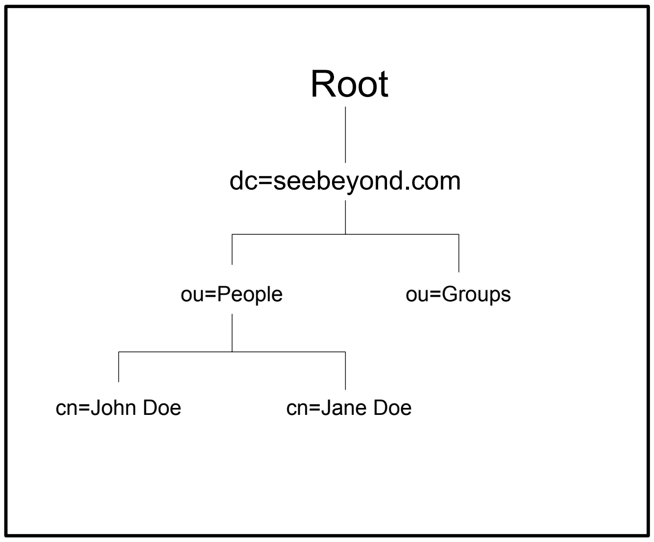
A DN is also a fully qualified path of names that trace the entry back to the root of the tree. For example, the distinguished name of the John Doe entry is:

```
cn=John Doe, ou=People, dc=seebeyond.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=seebeyond.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 4 on page 27 illustrates an example of an LDAP directory structure with distinguished names and relative distinguished names.

Figure 4 LDAP Directory Structure



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

4.1.4 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 One (formerly iPlanet) 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.

4.1.5 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 **"Referrals" on page 20**.

4.2 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:

- **"LDAP Root Node" on page 29**
- **"AddEntry Node" on page 29**
- **"STCEntry Subnode" on page 31**
- **"CompareEntry Node" on page 31**
- **"ModifyEntry Node" on page 33**
- **"PersistentSearch Node" on page 35**
- **"RemoveEntry Node" on page 38**
- **"RenameEntry Node" on page 38**
- **"Search Node" on page 39**
- **"TimestampSearch Node" on page 48**
- **"TlsExtension Node" on page 49**
- **"STCNotificationEvent Nodes" on page 49**
- **"Handling Exceptions" on page 50**

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

4.2.1 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.

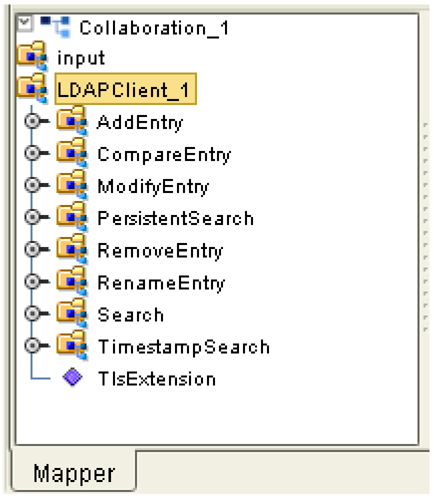
The LDAP OTD has the following components:

- 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 5 on page 29 shows an example of the LDAP OTD in the eGate Enterprise Designer's Collaboration Editor (Java) window.

Figure 5 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**. **Chapter 6** provides a general description of this document.

4.2.2 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.

4.2.3 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 6 shows the **AddEntry** node in its expanded form.

Figure 6 AddEntry Node

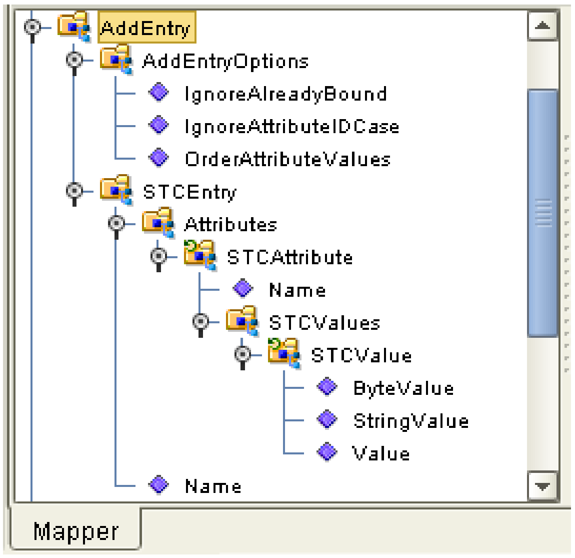


Table 2 explains the nodes and fields exposed on the **AddEntry** Node.

Table 2 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 3 on page 31**.

4.2.4 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 6 on page 30**.

See Table 3 for a description of this subnode, showing the subnode levels under the **STCEntry** subnode.

Table 3 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.

4.2.5 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 7 shows the **CompareEntry** node in its expanded form.

Figure 7 CompareEntry Node

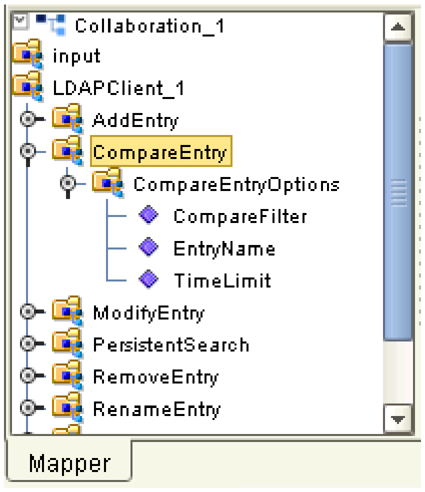


Table 4 explains the nodes and fields exposed on the **CompareEntry** node.

Table 4 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

4.2.6 ModifyEntry Node

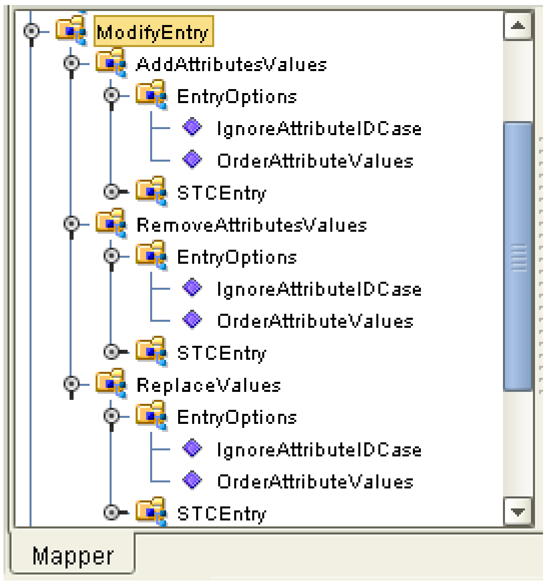
You can use the **ModifyEntry** node to modify an existing entry in a directory. You can make the following modifications to an entry:

- 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 8 shows the **ModifyEntry** node in its expanded form.

Figure 8 ModifyEntry Node



```
ModifyEntry
  AddAttributesValues
    EntryOptions
      IgnoreAttributeIDCase
      OrderAttributeValues
    STCEntry
  RemoveAttributesValues
    EntryOptions
      IgnoreAttributeIDCase
      OrderAttributeValues
    STCEntry
  ReplaceValues
    EntryOptions
      IgnoreAttributeIDCase
      OrderAttributeValues
    STCEntry
```

In Figure 8, 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 3 on page 31** for details on these subnodes.

Table 5 explains the nodes and fields exposed on the **ModifyEntry** node.

Table 5 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 5 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 3 on page 31**.
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.
performReplaceValuesmethod	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.

4.2.7 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 48**.

Figure 9 shows the **PersistentSearch** node in its expanded form.

Figure 9 PersistentSearch Node

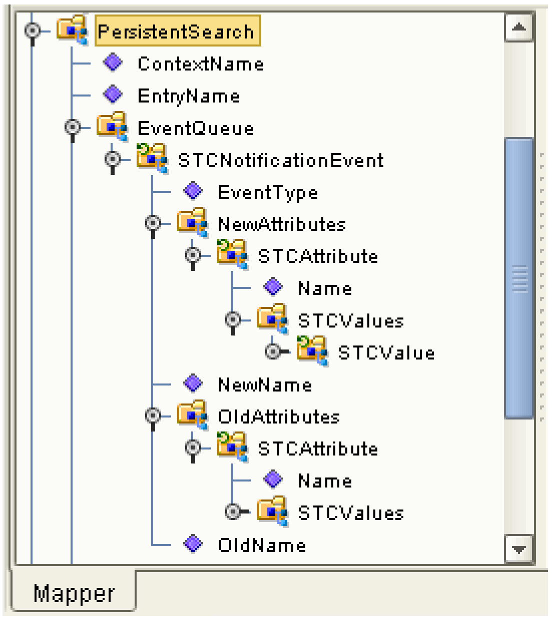


Figure 9 shows that the subnodes under **NewAttributes** and **OldAttributes** are identical in form to the subnodes under the **STCEntry** subnode. They also function in the same way. See **Table 3 on page 31** for details on these subnodes.

Persistent Search Limitations

The Persistent Search control feature has the following limitations:

- Works only against the Sun ONE server as Open LDAP and Active Directory.

  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 49** for details on this node.

The **PersistentSearch** node consists of subnodes as shown in Table 6.

Table 6 PersistentSearch Node

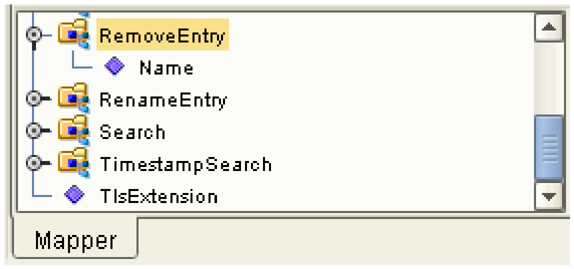
Name	Description
▪ **ContextName** field	Used to set the root of the search in the directory. The context name is relative to the context specified in the eWay's **ProviderURL** property. If the context name is not set correctly, the eWay is not able to properly resolve the context name relative to the 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.
SearchScope	The scope or boundary of the search. See **"SearchOptions Scopes" on page 43** for details.
EventQueue node	This node contains the results returned as an **STCNotificationEvent** node object (see **Table 14 on page 50** for details on this node's structure).
SearchFilter	The search filter expression per RFC 2254 to filter out the search results.

4.2.8 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 10 shows the **RemoveEntry** node in its expanded form.

Figure 10 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.

4.2.9 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 11 shows the **RenameEntry** node in its expanded form.

Figure 11 RenameEntry Node

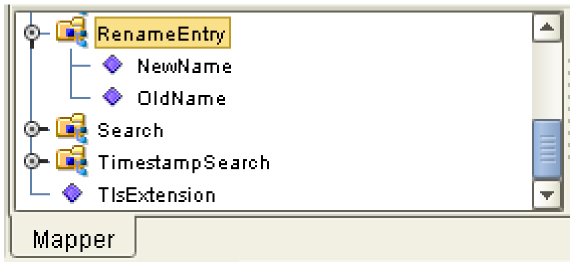


Table 7 describes the fields exposed on the **RenameEntry** node.

Table 7 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.

4.2.10 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 12 shows the **Search** node in its expanded form.

Figure 12 Search Node

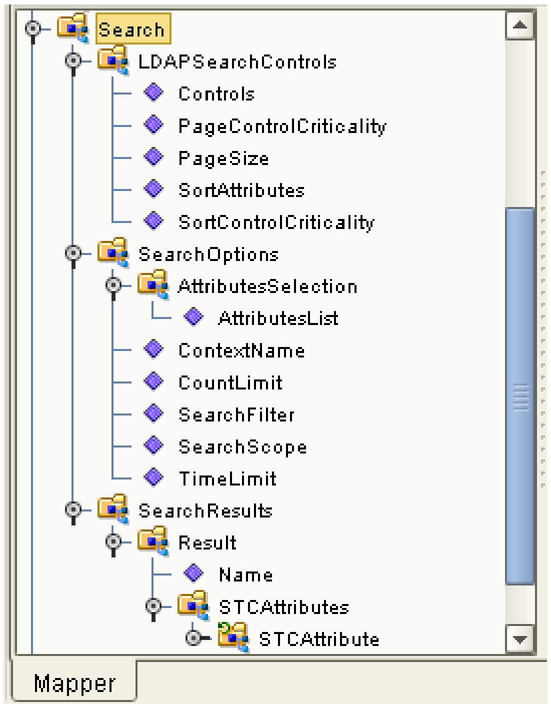


Figure 12 shows that the subnodes under **Result** are identical in form to the subnodes under the **STCEntry** subnode. They also function in the same way. See **Table 3 on page 31** for details on these subnodes.

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 36** *for details.*

Table 8 explains the fields exposed on the **LDAPSearchControls** node.

Table 8 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 9 explains the fields exposed on the **SearchOptions** node.

Table 9 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 45 and "SearchFilter Boolean Operators" on page 45. ***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 43 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.

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 10 for the appropriate OTD field node.

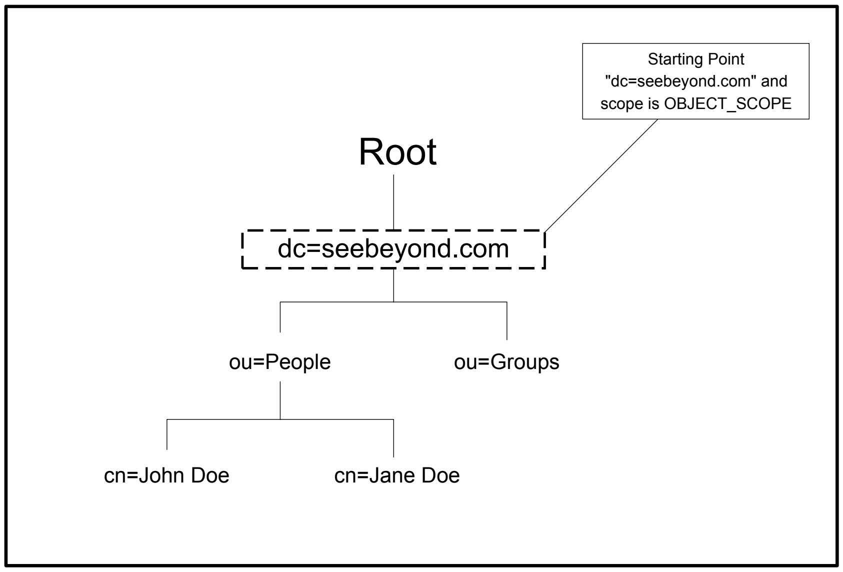
Table 10 Search Options Scope Parameters

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

The following list explains these parameters:

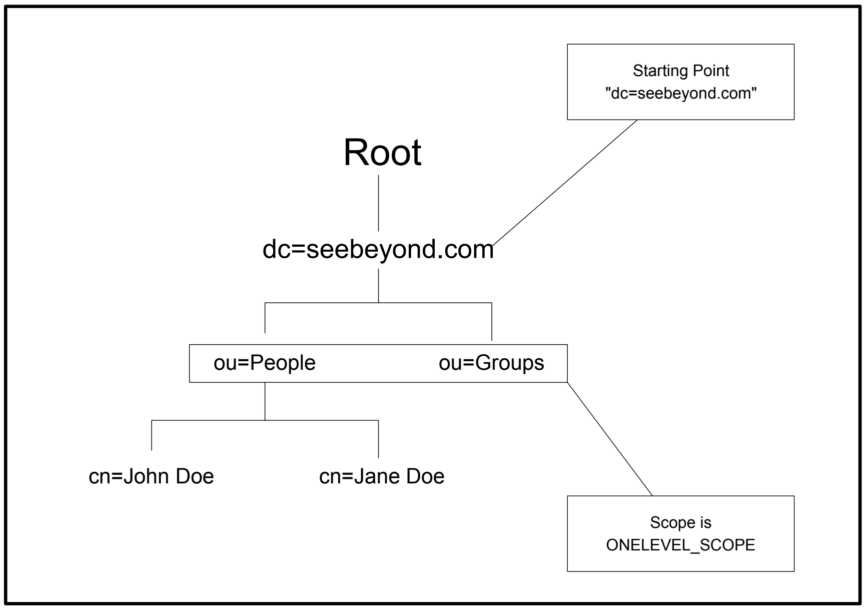
- **OBJECT_SCOPE** tells the eWay to search only within the named object, defined with **ContextName**. Using this scope essentially compares the named object for some particular attribute and/or value. See Figure 13.

Figure 13 OBJECT_SCOPE Diagram



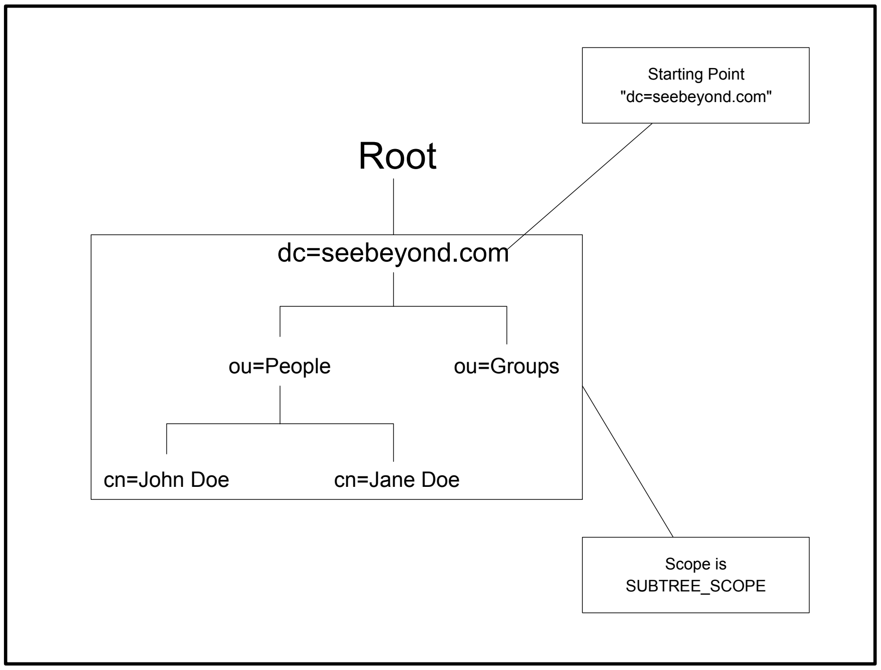
- **ONELEVEL_SCOPE** tells the eWay to search for entries one level below the named object. See **Figure 14 on page 44**.

Figure 14 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 15.

Figure 15 SUBTREE_SCOPE Diagram



SearchFilter Binary Operators

Binary operators that can be used in a filter expression are listed in Table 11.

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 11 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 12.

Table 12 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

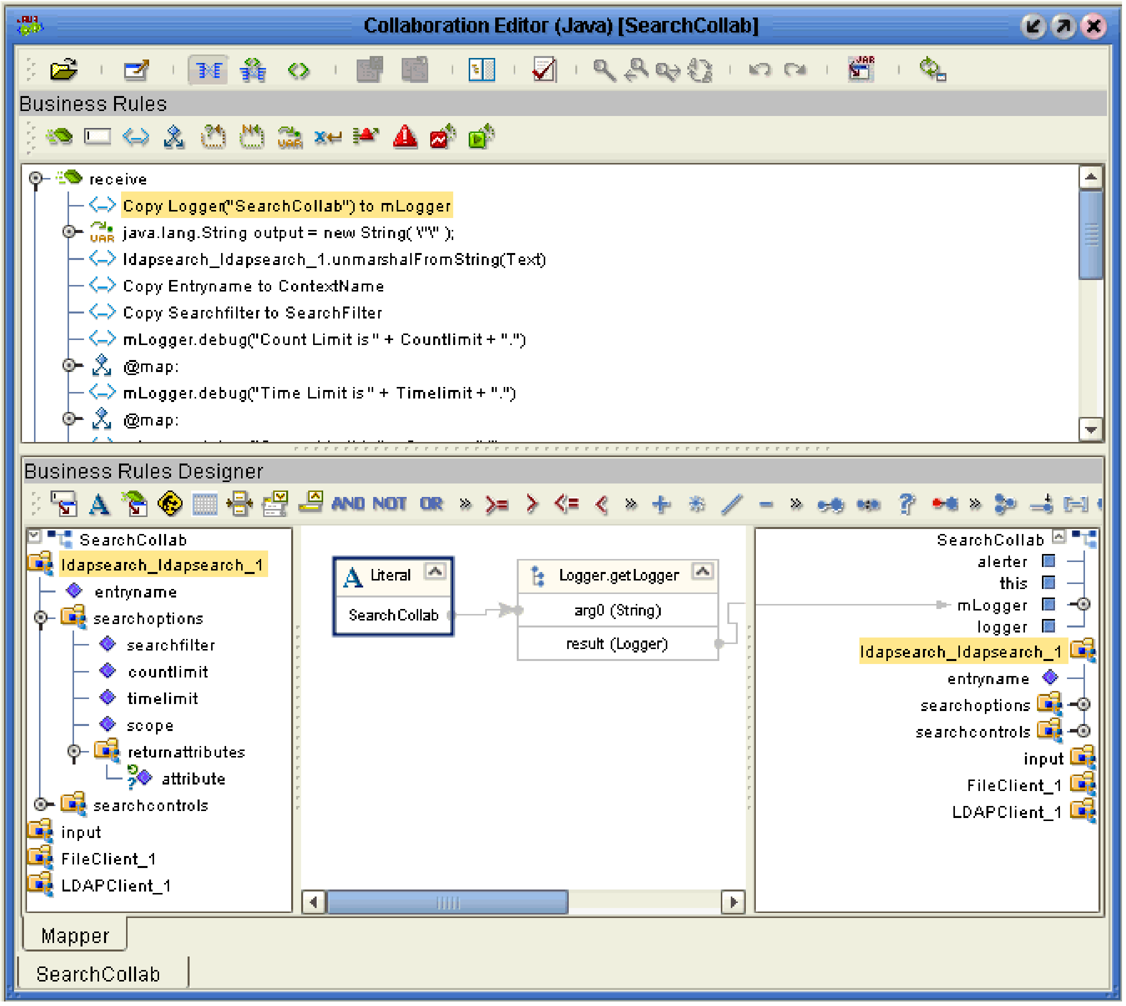
1. Call the **hasResults** method, which returns **true** if any results are returned, or **false** otherwise.
2. To iterate through all the entries, call **hasMoreResults** and **nextResult** within a **while** loop.

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 16 on page 47** for a Java-based Collaboration Definition from a Project sample that illustrates the use of the **SearchResults** operation.

Figure 16 SearchResults Operation in Collaboration Editor (Java)



```
receive
    Copy Logger("SearchCollab") to mLogger
    java.lang.String output = new String( "\"" );
    ldapsearch_ldapsearch_1.unmarshalFromString(Text)
    Copy Entryname to ContextName
    Copy Searchfilter to SearchFilter
    mLogger.debug("Count Limit is " + Countlimit + ".")
    @map:
    mLogger.debug("Time Limit is " + Timelimit + ".")
    @map:
```

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

```
System.out.println ("Entry>>>> " + LDAPClient_1.getSearch().
    getSearchResults().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 3 on page 31 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:

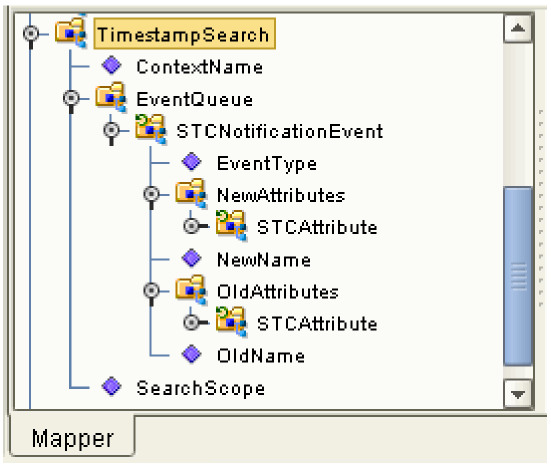
```
if(LDAPClient_1.getSearch().getSearchResults().getResult().
    getSTCAttribute(i).getSTCValue(j).isString())
             {
                System.out.println ("    Value [String]: " +
    getLDAP().getSearch().getSearchResults().getResult().
    getSTCAttribute(i).getSTCValue(j).getStringValue());
             }
```

4.2.11 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 35**). The node's operation uses the timestamp of the entries to track the updates.

Figure 17 shows the **TimestampSearch** node in its expanded form.

Figure 17 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 Open LDAP 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 49** for details on this node.

The **TimestampSearch** node consists of subnodes as shown in Table 13.

Table 13 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 43** for details.
EventQueue node	This node contains the results returned as an **STCNotificationEvent** object (see **Table 14 on page 50** for details on this node's structure).

4.2.12 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 17** for details and an example.

4.2.13 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 14.

Table 14 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** and **Chapter 6**.*

4.3 Handling Exceptions

In using the LDAP OTD, you may encounter the following exception specific to this eWay:

com.stc.connector.appconn.ldap.LDAPApplicationException

To handle this exception, you can use the following **.jar** file:

stcldap14.jar

Import this file as follows:

1 Click the current Project in the Enterprise Designer's **Project Explorer**.

2 Right-click the Project and choose **New > File** from the shortcut menu.

A dialog box appears, which prompts you to choose a file.

3 Navigate to the **ldapadapter.jar** file at the following directory:

<ICAN_install_directory>\edesigner\ussrdir\modules\ext\ldapadapter

4 Select **ldapadapter.jar** and click **Import**.

The name of the selected file appears under the current Project in the **Project Explorer**.

5. Extract **stcldap14.jar** from the **ldapadapter.jar** file.
6. Create a new Collaboration Definition for the purpose of handling LDAP eWay-specific exceptions.
7. Using the **JAR File Import** feature in the Collaboration Editor (Java), add **stcldap14.jar** to this Collaboration Definition.
8. Use the Collaboration Editor (Java) as desired to create the needed exception handling.

   You can use this Collaboration Definition and this file to create Business Rules that handle this exception.

See the *eGate Integrator User's Guide* for details on how to use the Collaboration Editor (Java).

Chapter 5

Reviewing the Sample Project

This chapter describes how to implement the LDAP eWay by way of reviewing the sample Project included with the eWay. The chapter assumes that you are already familiar with eGate concepts and that you understand the basics of creating a Project using the eGate Enterprise Designer.

For a complete explanation of eGate terminology and concepts, including how to use the eGate Enterprise Designer to create and set properties for the components of an eGate Project, see the *eGate Integrator User's Guide*.

Chapter Topics

5.1 eGate Project Description

This section provides an overview of the eGate sample Project for the LDAP eWay and how to import the Project into eGate.

5.1.1 Projects and the Enterprise Designer

A Project contains all of the eGate components you designate to perform one or more desired processes in eGate. Each eGate Project is created using the Enterprise Designer.

The Project canvas contains windows that represent the various stages of your Project. The types of windows in your Project canvas area include:

- **Connectivity Map Canvas**: Contains the eGate business logic components, such as Collaborations, Topics, Queues, and eWays, that you include in the structure of the Project.
- **OTD Editor**: Contains the source files used to create Object Type Definitions (OTDs) to use with a Project.
- **Collaboration Editor (Java)**: Allows you to create and/or modify Business Rules to implement the business logic of a Project's Java-enabled Collaboration Definitions.

5.1.2 Importing Sample Projects

Before you can view or work with a sample Project, you must first import it into eGate, using the Enterprise Designer.

Note: *The sample .**zip** file you first download may contain more than one Project and/or additional files. If this is the case, you must unzip this file first, find the desired Project file, then import the Project file. For information on how to obtain this file, see* **Chapter 2**. *For complete instructions on downloading and installing the eWay and related files, see the **SeeBeyond ICAN Suite Installation Guide**.*

The container file you are looking for is **LDAP_eWay_Sample.zip**. The Project file name is:

LDAP_SampleProject.zip: Java-based Collaboration Project

You can name imported Projects as desired.

To import a sample Project

1. Save any changes not saved previously.
2. From the Enterprise Designer's **Project Explorer** pane, right-click the desired Repository and select **Import**.
3. On the **Import Manager** window, browse to the directory that contains the sample Project **.zip** file.
4. Select the sample Project file and click **Open**.
5. Click **Import**. If the import was successful, click **OK** on the **Import Status** dialog box.
6. Close the **Import Manager** window.

Important: *An imported Project does not contain an environment or a deployment profile. After importing a Project, you must use the Enterprise Designer to create these functions for the Project. See* **"Creating the Project's Environment" on page 60** *and* **"Deploying a Project" on page 61**. *For additional information, see the **eGate Integrator User's Guide** and **SeeBeyond ICAN Suite Deployment Guide**.*

You must check out the major eGate components before you can change them. For details, see the *eGate Integrator User's Guide*.

5.1.3 Basic eWay Components

The sample Project allows you to see how to set up the eWay to perform operations within eGate. To use the LDAP eWay in a Project, you must also set the desired eWay properties.

LDAP eWay Properties

The properties for the LDAP eWay contain the settings used to connect with a specific external system. You can change these settings using the eGate eWay **Properties** sheet. For more information about LDAP eWay properties and this dialog box, see **Chapter 3**.

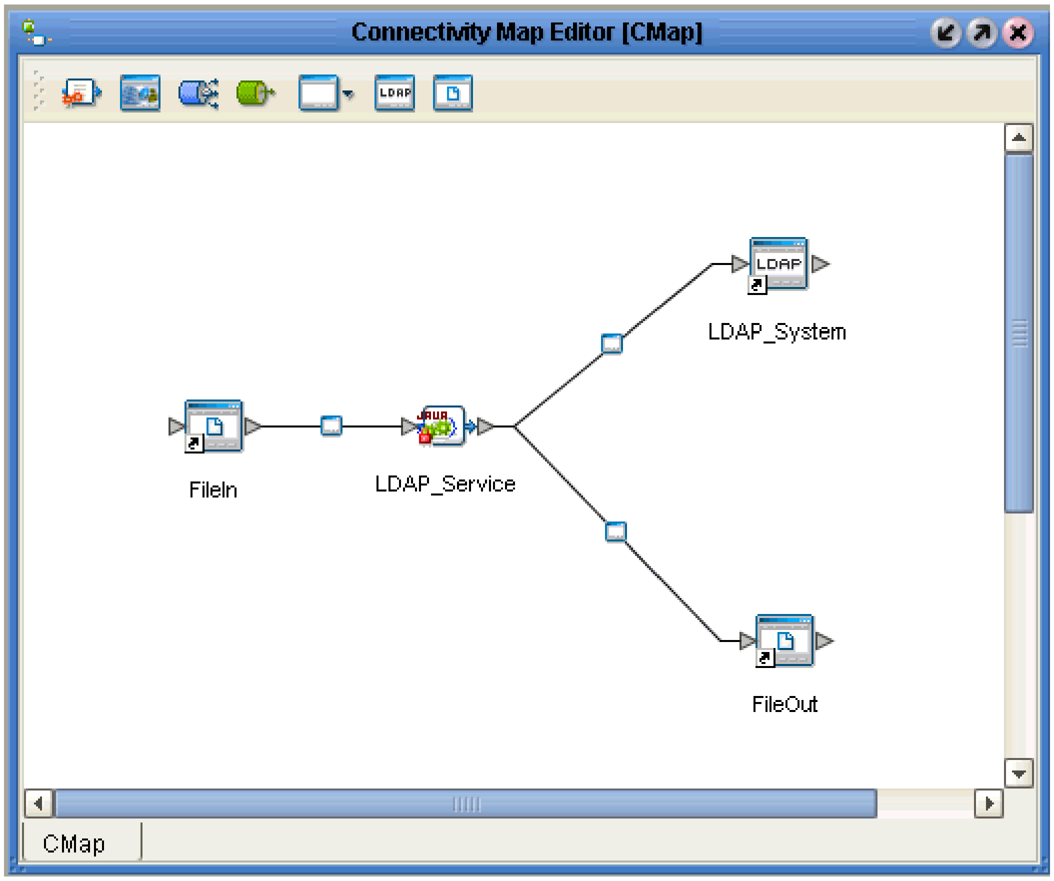
LDAP OTD

A versatile LDAP OTD comes packaged with the eWay. This OTD contains eWay properties and methods and also allows you to utilize, as well as extend, the eWay's features. See **Chapter 4** for details.

5.1.4 Overview: Sample Project

The LDAP eWay sample Project demonstrates how the LDAP eWay processes information from a LDAP system. The resulting information is then written to a text file. **Figure 18 on page 55** shows a general diagram of how this scenario operates, using the eGate Connectivity Map.

Figure 18 Sample Project Scenario



Project Components

The Project has the following components:

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

Project Operation

The Project operates as follows:

- **FileIn**: The external file system that 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**: 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 system; the LDAP eWay handles inbound and outbound communication with this system.
- **FileOut**: The external file system that receives the query information; another File eWay writes the received information to a text file on this system.

Input Data

When you extract the sample Project file, you get an additional **.zip** file, **input_data.zip**. The **input_data.zip** file contains files you can use as input data for the Project.

File Locations

When you unzip **input_data.zip**, you get a folder named **Data**. This folder contains a set of folders named for the LDAP operation to be performed (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.

Table 15 shows the sample data folders and the **.dtd** files they contain.

Table 15 Project Input Data

Folder Name	DTD File Name
AddEntry	ldapadd.dtd
SearchEntry	ldapsearch.dtd, ldapssl.dtd, and ldapstarttls.dtd
CompareEntry	ldapcompare.dtd
ModifyEntry	ldapmodify.dtd
RenameEntry	ldaprenam.dtd
RemoveEntry	ldapremove.dtd
PersistentSearch	ldappersistentsearch.dtd
TimestampSearch	ldaptimestampsearch.dtd

Naming Convention for .xml Files

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**: Open LDAP
- **SO**: Sun ONE

For example, the **input_ldapmodify_replaceattr_SO.xml** file is for the modify operation to replace an attribute and 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.

5.2 Summary: Sample Collaboration (Java) Project

This section explains generally how to implement the LDAP eWay using the eGate Project sample that includes a Java-based Collaboration. This sample is included on your installation CD-ROM.

The extracted sample Project file is named **LDAP_SampleProject.zip**. This Project allows you to observe an end-to-end data-exchange scenario involving eGate and the LDAP eWay.

For instructions on how to import the sample Project, see the **procedure on page 53**. For an overview of the sample Project and what is does, see **"Overview: Sample Project" on page 54**.

See the *eGate Integrator User's Guide* for details on how to build an end-to-end Project in eGate.

5.2.1 Using the eWay With Java-based Collaborations

Java-based Collaborations in eGate use the LDAP OTD to process LDAP data within the eGate system. Using Collaboration Definitions and these OTDs, you can allow the eWay to access the desired LDAP functions.

See **Chapter 4** for details on the LDAP OTD.

Creating Business Rules Within Collaboration Definitions

A Collaboration Editor (Java) allows you to create the Business Rules that implement your business logic for a Java-based Collaboration Definition.

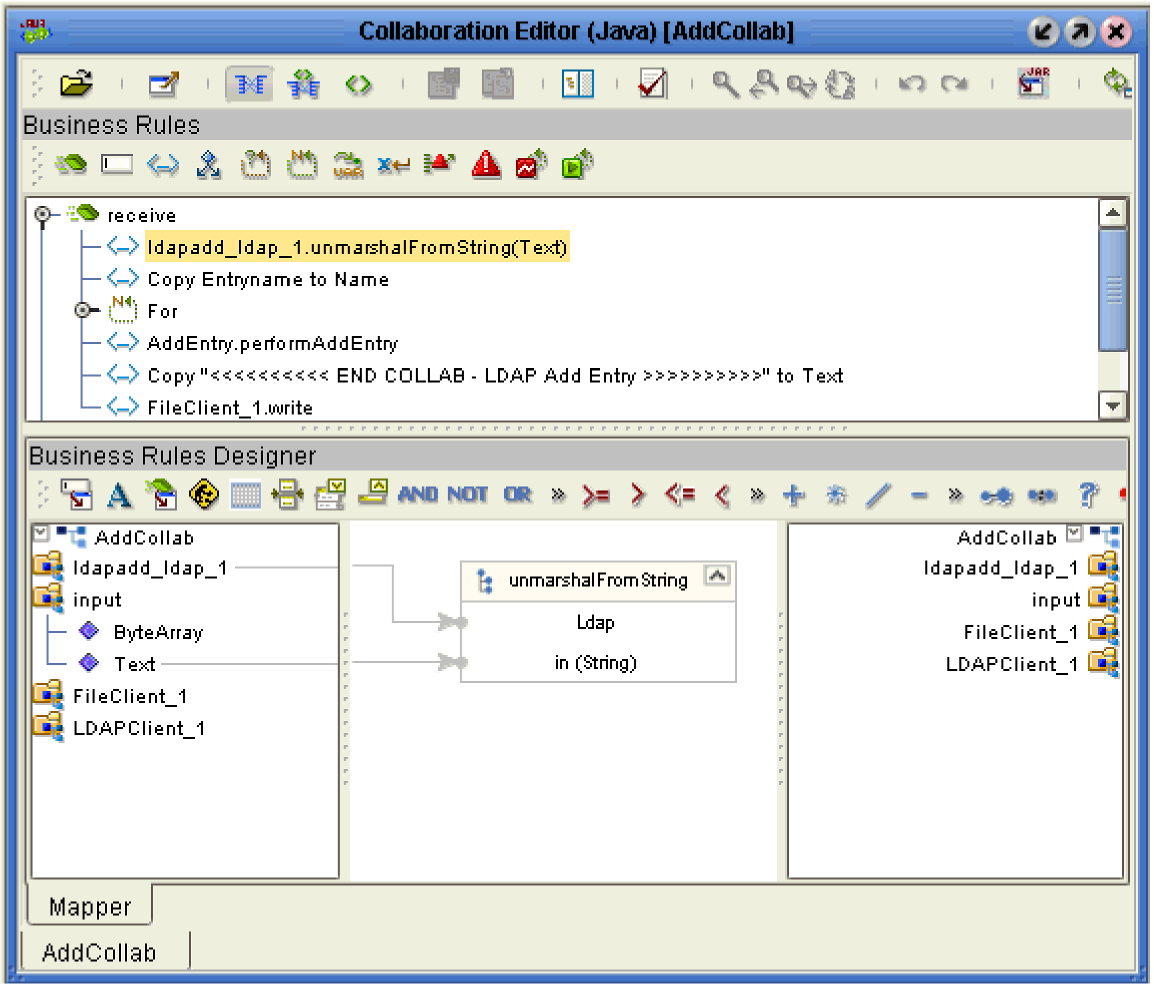
See the *eGate Integrator User's Guide* for complete information on how to use the Collaboration Editor (Java).

Collaboration Editor (Java) Window

The Collaboration Editor (Java) window displays in the Enterprise Designer after you create a Java-based Collaboration Definition. You can also open this editor by right-clicking on the name of the desired Collaboration Definition in the **Project Explorer** and choosing **Open** from the shortcut menu.

To complete a Java-based Collaboration Definition, you can use this editor to create Business Rules. See Figure 19 for an example of the Collaboration Editor (Java).

Figure 19 Collaboration Editor (Java) and Business Rules



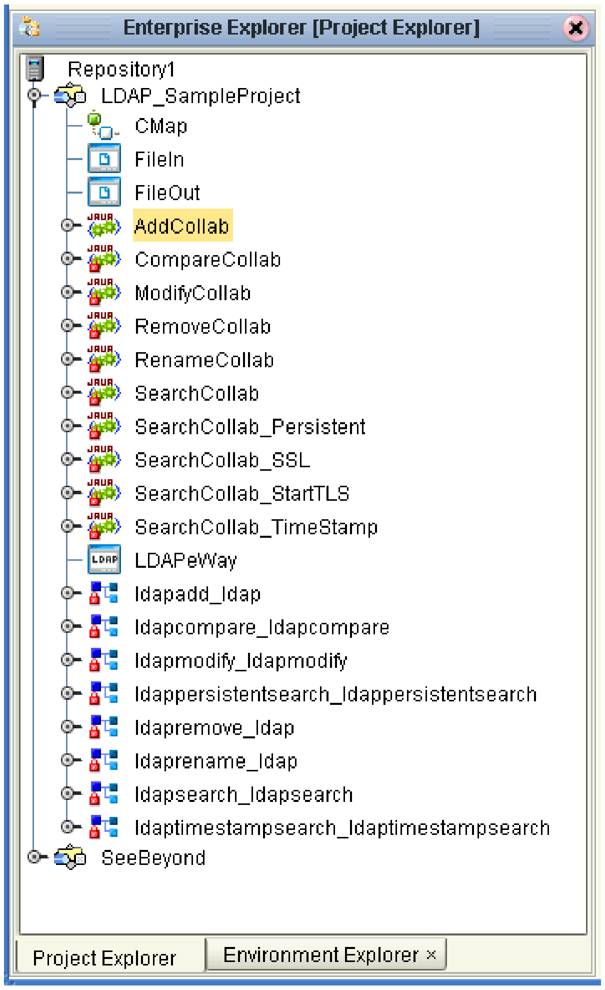
ldapadd_ldap_1.unmarshalFromString(Text)
Copy Entryname to Name
For
AddEntry.performAddEntry
Copy "<<<<<<<<<< END COLLAB - LDAP Add Entry >>>>>>>>>>" to Text
FileClient_1.write

You can create the desired Business Rules for your Project by dragging and dropping values from a source OTD onto the nodes of a destination LDAP OTD and other OTDs. LDAP OTD nodes represent LDAP functions, which are in turn able to call LDAP methods.

Collaboration Definitions in Sample

The sample comes with Java Collaboration Definitions already built for you. These Collaboration Definitions model the basic LDAP operations enabled by the LDAP OTD. See Figure 20 the list of sample Collaboration Definitions and OTDs as shown in the Project Explorer.

Figure 20 Sample Collaboration Definitions and OTDs



Enterprise Explorer [Project Explorer]

```
Repository1
  LDAP_SampleProject
    CMap
    FileIn
    FileOut
    AddCollab
    CompareCollab
    ModifyCollab
    RemoveCollab
    RenameCollab
    SearchCollab
    SearchCollab_Persistent
    SearchCollab_SSL
    SearchCollab_StartTLS
    SearchCollab_TimeStamp
    LDAPeWay
    ldapadd_ldap
    ldapcompare_ldapcompare
    ldapmodify_ldapmodify
    ldappersistentsearch_ldappersistentsearch
    ldapremove_ldap
    ldaprename_ldap
    ldapsearch_ldapsearch
    ldaptimestampsearch_ldaptimestampsearch
  SeeBeyond
```

The Collaboration Definitions and OTDs are named according to the operation or operations they perform. For example, **CompareCollab** does a *compare* operation. To enable a Service to perform a given operation, drag and drop the desired Collaboration Definition onto a service on the Connectivity Map.

For details on these operations and how they work using the LDAP OTD, see **Chapter 4**.

5.2.2 Completing the Project

For instructions on how to finish implementing your Project, see the following sections:

- **"Creating the Project's Environment" on page 60**
- **"Setting eWay Properties" on page 60**
- **"Deploying a Project" on page 61**

5.3 Creating the Project's Environment

This section provides general procedures for creating an Environment for your Project. For a complete explanation, see the *eGate Integrator User's Guide.*

To create an Environment

1. From the Enterprise Designer, click the **Environment Explorer** tab on the Enterprise Explorer.
2. Under the current **Repository** icon in the **Environment Explorer**, create a new Environment for your Project and name it as desired.
3. In the **Environment Explorer**, right-click the **Environment** icon and select the desired external systems from the shortcut menu. Include external systems for the LDAP eWay and the File eWays (inbound and outbound). Give them the same names as you did the corresponding external applications on the Connectivity Map.
4. Use the same shortcut menu to create a Logical Host for your Project, and name it as desired.
5. Click **Save** and return to the **Project Explorer** tab.

5.4 Setting eWay Properties

You must set the eWay properties for your specific system and for the current Project, using the eGate Enterprise Designer. For directions on accessing and using the eWay **Properties** sheet, as well as a complete explanation of the LDAP eWay properties, see Chapter 3.

To set properties for the File eWays

1. From the **Project Explorer**, open the **CMap** Connectivity Map for the sample Project.
2. To change the default properties for the inbound File eWay, double-click the **FileIn** external system's eWay icon. For this eWay, select the **Inbound** properties.

   The eWay **Properties** sheet appears. This eWay can use the current default properties settings for the sample Project. However, if you are using different file names and/or folders than the defaults, you must enter the names of the files/ folders you are using.

Note: *Even if you do not change the eWay's properties, you must open each **Properties** sheet for every eWay and click **OK** to activate the eWay.*

3. For this sample, use the **C:\temp** as the property for **Directory**, and for **Input file name**, enter ***.txt**.
4. Click **OK** to save the settings and close the eWay **Properties** sheet.
5. To change the default properties for the outbound File eWay, double-click the **FileOut** external system's eWay icon. For this eWay, select the **Outbound** properties.

   Use your file and folder names if they differ from the defaults. For all other properties, you can use the defaults.
6. Click **OK** to close the properties sheet and save your changes.

To set properties for the LDAP eWays

1. To begin changing the default properties for the LDAP eWay, double-click the **LDAPeWay** external system's eWay icon.

   The eWay **Properties** sheet appears.
2. The properties settings appear in the **Properties** pane on the left.
3. Set the properties as desired then click **OK** to close the dialog box and save.

   See **Chapter 3** for details on how to set LDAP eWay properties.

5.5 Deploying a Project

This section provides general procedures for Project deployment.

5.5.1 Basic Steps

For a complete explanation of how to deploy and run an eGate Project, see the *eGate Integrator User's Guide.*

To deploy the Project

1. From the **Project Explorer**, select the current Project and right-click, choosing **New > Deployment Profile** from the shortcut menus.
2. From the **Create a Deployment Profile** dialog box, enter the name of the current Project and select the Environment you created for this Project.
3. Click **OK**.

The Deployment Profile canvas appears as follows:

- The Project's external applications and Services show up as items on the left side of the canvas.
- The external systems and Logical Host you created under **"Creating the Project's Environment" on page 60** show up as windows on the right side of the canvas.

4. Set up your Deployment Profile by dragging the items on the left into the corresponding window on the right.
5. Click **Save All** then **Activate**.

   When the Project has been activated, a pop-up message appears stating the activation was successful.

For additional information, see the *SeeBeyond ICAN Suite Deployment Guide.*

To run the Project

For instructions on how to run a Project, see the *eGate Integrator Tutorial*.

5.5.2 Alerting and Logging

eGate provides an alerting and logging feature. This feature allows monitoring of messages and captures any adverse messages in the order of severity, based on a configured severity level and higher. For details on how to enable the Logging feature, see the *eGate Integrator User's Guide*.

Chapter 6

Using eWay Java Methods

This chapter provides an overview of the Java classes and methods contained in the LDAP eWay. These methods are used to extend the functionality of the eWay.

Chapter Topics

6.1 LDAP eWay Methods: Overview

The LDAP eWay exposes various Java methods to add extra functionality to the eWay. These methods make it easier to set information in the LDAP eWay Object Type Definitions (OTDs), as well as get information from them.

6.1.1 Relation to eWay Properties

The nature of this data transfer depends on the properties you set for the eWay in the eGate Enterprise Designer. For more information on the eWay's properties, see **Chapter 3**.

6.1.2 LDAP eWay Javadoc

For a complete list of the Java methods within the classes listed in this chapter, refer to the **Javadoc**. You can download the Javadoc while you are installing the eWay. The download file is named **LDAP_eWay_Javadoc.zip**. For more information on how to download this file, see **Chapter 2**.

For complete instructions on downloading and installing the eWay and related files, see the *SeeBeyond ICAN Suite Installation Guide*.

6.2 eWay Java Classes

The LDAP eWay provides the following Java classes:

- **AddAttributesValues**
- **Add**
- **AddEntry**
- **AddEntryOptions**
- **AttributesSelection**
- **CompareEntry**
- **CompareEntryOptions**
- **EntryOptions**
- **LDAPApplicationException**
- **LDAPClientApplication**
- **LDAPClientApplicationImpl**
- **LDAPSearchControls**
- **ModifyEntry**
- **PersistentSearch**
- **RemoveAttributesValues**
- **RemoveEntry**
- **RenameEntry**
- **ReplaceValues**
- **Result**
- **Search**
- **SearchOptions**
- **SearchResults**
- **StartTLSExtension**
- **STCAttribute**
- **STC**
- **STCAttributes**
- **STCEntry**
- **STCEventQueue**
- **STCNotificationEvent**

- **STCValue**
- **STCValues**
- **TimestampSearch**

Index