SeeBeyond™ eBusiness Integration Suite

e*Way Intelligent Adapter for LDAP User's Guide

Release 4.5.3

Java Version

Version 20021113084759.

Contents

Chapter 1

Introduction

This document describes how to install and configure the Java version of the e*Way Intelligent Adapter for LDAP (Lightweight Directory Access Protocol), version 3.0.

1.1 Intended Reader

The reader of this guide is presumed:

- to be a developer or system administrator with the responsibility of maintaining the e*Gate system.
- to have high-level knowledge of Windows or UNIX operations and administration.
- to have high-level knowledge of LDAP directory services.
- to be familiar with *Transmission Control Protocol/Internet Protocol* (TCP/IP).
- to be thoroughly familiar with Windows-style GUI operations.

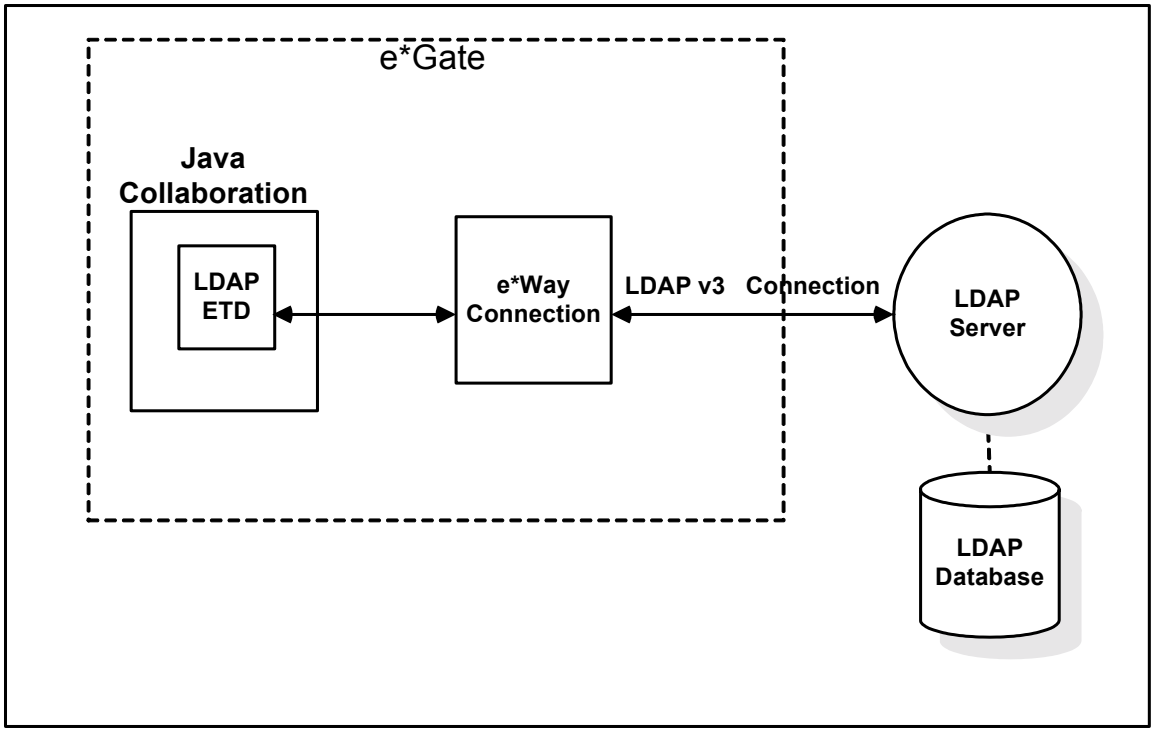
1.2 Overview

1.2.1 e*Way Intelligent Adapter for LDAP

The LDAP e*Way enables e*Gate to exchange data with an LDAP directory. It consists of two separate components: the LDAP Connector and the LDAP ETD (Event Type Definition). The LDAP ETD utilizes the LDAP Connector to connect to a particular LDAP server. An instance of an LDAP ETD utilizes only one instance of an LDAP Connector.

In addition, the LDAP ETD exposes the API for accessing the LDAP directory. The LDAP ETD enables a user to write Java collaboration rules to execute operations such as searching the directory, adding entries to a directory, and modifying entries in the directory. See Figure 1 for the architecture of the LDAP e*Way.

Figure 1 Architecture of the LDAP e*Way:



By connecting to an LDAP server, the LDAP e*Way provides a powerful addition to the e*Gate platform by enabling e*Gate to search, compare, and modify an LDAP directory using the LDAP protocol. For example, the LDAP e*Way, in conjunction with e*Gate, can be used 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 the appropriate permissions can access thousands of employee records.

Note: The *LDAP e*Way* supports subcollaborations. *For further information on using Subcollaboration Rules see the* ***e*Gate Integrator User's Guide****.*

Sample schemas for the Java enabled LDAP e*Way are included on the installation CD-ROM which demonstrate how simple scenarios are managed.

The following sections discuss LDAP and the e*Way in further detail.

1.2.2 LDAP

LDAP is a directory service protocol that runs over TCP/IP. A directory service is a distributed database application designed to manage the entries and attributes in a directory. LDAP allows clients to access different directory services based on entries, and makes the entries and their attributes and values available to users and other applications.

Entries, Attributes and Values

An LDAP directory has entries which 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 email 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, mail, 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 example above, the mail attribute contains two mail values.

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 the smaller organizations that compose the larger ones. The leaf nodes (or entries) of the tree structure represent the individual people or resources.

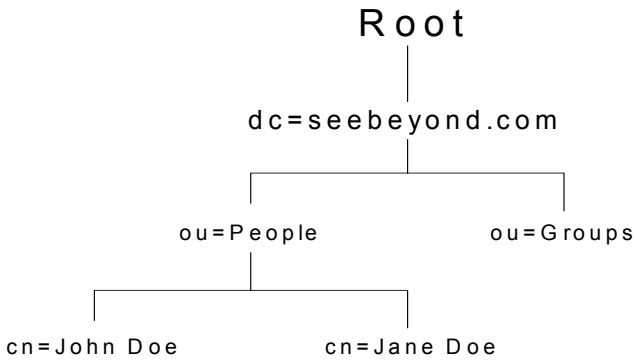
Distinguished Names and Relative Distinguished Names

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 (or 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 2 illustrates an example of an LDAP directory structure with distinguished names and relative distinguished names.

Figure 2 LDAP Directory Structure



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 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.

1.2.3 Java Naming and Directory Interface (JNDI)

The LDAP e*Way uses Sun's Java Naming and Directory Interface™ (JNDI) LDAP provider. JNDI is an API published by Sun. This set of APIs allows a Java program to store objects and lookup objects using multiple naming services in a standard manner.

The JNDI is included in the Java 2 SDK version 1.3 that is installed as part of e*Gate.

Examples of other available APIs include Netscape® Directory SDK (C and Java software libraries), and Novell® Java LDAP (JLDAP). These are not used or supported by the LDAP e*Way.

1.2.4 Referrals

The native APIs developed for the LDAP e*Way 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 which holds the actual entry. The LDAP URL contains the server's host/port and an object's DN.

For e*Way connection configuration for referrals, see **Referrals** on page 27.

1.3 LDAP e*Way Components

The LDAP e*Way is comprised of the following components:

- **Jar files** containing the logic required by the e*Way to gain access to LDAP:
  - **stcnsclient.jar**, the executable component.
  - **stcnsconnector.jar**, contains the classes that implement the LDAP connector.
- The **LDAP e*Way Connection** which provides access to information necessary for connecting to specified external LDAP server connections.

A complete list of installed files appears in **Table 1 on page 16**.

1.4 Supported Operating Systems

The LDAP e*Way is available on the following operating systems:

- Windows XP
- Windows 2000, Windows 2000 SP1, Windows 2000 SP2, and Windows 2000 SP3
- Windows NT 4.0 SP6a
- Solaris 2.6, 7, and 8
- HP-UX 11.0 and HP-UX 11i

1.5 System Requirements

To use the LDAP e*Way, you need the following:

- An e*Gate Participating Host, version 4.5.1 or later. For Windows XP operating system, you need an e*Gate Participating Host, version 4.5.3 or later.
- A TCP/IP network connection
- A computer running Windows, to allow you to use the e*Gate Enterprise Manager and ETD Editor
- Additional disk space for e*Way executable, configuration, library, and script files. The disk space is required on both the Participating and the Registry Host. Additional disk space is required to process and queue the data that this e*Way

processes. The amount necessary varies based on the type and size of the data being processed, and any external applications performing the processing.

- Open and review the **Readme.txt** for the LDAP e*Way for any additional requirements prior to installation. The **Readme.txt** is located on the Installation CD_ROM at setup\addons\ewldap.

Installed on the Participating Host

- Java JDK version 1.3.1

The e*Way must be configured and administered using the Enterprise Manager.

1.6 External System Requirements

- LDAP server supporting LDAP Version 3.0

Note: *LDAP Version 2 is not currently supported by the e*Way Intelligent Adapter for LDAP. However, relevant operations for LDAP Version 2 servers will perform satisfactorily with the e*Way.*

To enable the e*Way to communicate properly with the LDAP system, the following are required:

- 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

1.7 Supporting Documents

The following SeeBeyond documents are designed to work in conjunction with the *e*Way Intelligent Adapter for LDAP User's Guide* and to provide additional information that may prove useful to you:

- *Creating an End-to-end Scenario with e*Gate Integrator*
- *e*Gate Integrator Alert Agent User's Guide*
- *e*Gate Integrator Alert and Log File Reference Guide*
- *e*Gate Integrator Collaboration Services Reference Guide*
- *e*Gate Integrator Installation Guide*
- *e*Gate Integrator Intelligent Queue Services Reference Guide*
- *e*Gate Integrator SNMP Agent User's Guide*
- *e*Gate Integrator System Administration and Operations Guide*
- *e*Gate Integrator User's Guide*
- *Standard e*Way Intelligent Adapters User's Guide*
- *Readme.txt* file on the e*Gate installation CD-ROM.

Chapter 2

Installation

This chapter describes the procedures for installing the LDAP e*Way.

2.1 Installing the LDAP e*Way on Windows NT 4.0 and Windows 2000

2.1.1 Pre-installation

- Exit all Windows programs before running the setup program, including any antivirus applications.
- You must have Administrator privileges to install this e*Way.

2.1.2 Installation Procedure

To install the LDAP e*Way on a Windows system

1. Log in as an Administrator to the workstation on which you are installing the e*Way.
2. Insert the e*Way installation CD-ROM into the CD-ROM drive.
3. If the CD-ROM drive's Autorun feature is enabled, the setup application launches automatically; skip ahead to step 4. Otherwise, use the Windows Explorer or the Control Panel's **Add/Remove Applications** feature to launch the file **setup.exe** on the CD-ROM drive.
4. The InstallShield setup application launches. Follow the installation instructions until you come to the **Please choose the product to install** dialog box.
5. Select e*Gate Integrator, then click **Next**.
6. Follow the on-screen instructions until you come to the second **Please choose the product to install** dialog box.
7. Clear the check boxes for all selections except Add-ons, and then click **Next**.

8 Follow the on-screen instructions until you come to the **Select Components** dialog box.

9 Select (but do not check) **e*Ways**, and then click the **Change** button. The **SelectSub-components** dialog box appears.

10 Select the **LDAP e*Way**. Click the continue button to return to the Select Components dialog box, then click **Next**.

11 Follow the rest of the on-screen instructions to install the LDAP e*Way. Be sure to install the e*Way files in the suggested client installation directory. The installation utility detects and suggests the appropriate installation directory. Unless you are directed to do so by SeeBeyond support personnel, do not change the suggested installation directory setting.

Note: *Once you have installed and configured this e*Way, you must incorporate it into a schema by defining and associating the appropriate Collaborations, Collaboration Rules, IQs, and Event Types before this e*Way can perform its intended functions. For more information about any of these procedures, see the online Help.*

*For more information about configuring e*Ways or how to use the e*Way Editor, see the e*Gate Integrator User's Guide.*

2.2 Installing the LDAP e*Way on UNIX

2.2.1 Pre-installation

Root privileges are not required to install this e*Way. Log in under the user name that you wish to own the e*Way files. Be sure that this user has sufficient privileges to create files in the e*Gate directory tree.

2.2.2 Installation Procedure

To install the LDAP e*Way on a UNIX system

1 Log in on the workstation containing the CD-ROM drive, and insert the CD-ROM into the drive.

2 If necessary, mount the CD-ROM drive.

3 At the shell prompt, type

cd /cdrom

4 Start the installation script by typing

setup.sh

5 A menu of options will appear. Select the **Install e*Way** option. Then, follow the additional on-screen directions.

Note: *Be sure to install the e*Way files in the suggested* ***client*** *installation directory. The installation utility detects and suggests the appropriate installation directory.* ***Unless you are directed to do so by SeeBeyond support personnel, do not change the suggested "installation directory" setting.***

6 After installation is complete, exit the installation utility and launch the Enterprise Manager.

Note: *Once you have installed and configured this e*Way, you must incorporate it into a schema by defining and associating the appropriate Collaborations, Collaboration Rules, IQs, and Event Types before this e*Way can perform its intended functions. For more information about any of these procedures, see the online Help system.*

*For more information about configuring e*Ways or how to use the e*Way Editor, see the* ***e*Gate Integrator User's Guide****.*

2.3 Files/Directories Created by the Installation

The LDAP e*Way installation process installs the files shown in Table 1 within the e*Gate directory tree. Files are installed within the **egate\client** tree on the Participating Host and committed to the **default** schema on the Registry Host.

Table 1 Files Created by the Installation

e*Gate Directories	File(s)
\classes\	stcnsclient.jar stcnsconnector.jar
\etd\	stcewldap.ctl
\configs\ldap	ldap.def
\etd\ldap\	ldap.xsc
\ThirdParty\jndi\ldap1.2.4\classes	ldapbp.jar
\ThirdParty\gnu-getopt\classes	gnu-getopt.jar

The ldapbp.jar file

The LDAP e*Way includes the ldapbp.jar JAR file, which contains the necessary LDAP control classes used by the e*Way.

In addition, the ldapbp.jar JAR file contains the LDAP "booster" pack, which provides extensions to the LDAP service provider. The "booster" pack allows using server side controls and increases performance greatly. For example, when using a paging control, the client sends a page control request to the server such that the server can return results on a page-by-page basis when a search returns too many results. See **"LDAPSearchControls" on page 38** for more information about LDAP controls.

Note: *The LDAP service provider is included in the Java 2 SDK installed as part of e*Gate. See the e*Gate Integrator Installation Guide for more information on installing Java 2 SDK.*

Chapter 3

Configuration

This chapter describes how to configure the following components of the LDAP e*Way.

3.1 Multi-Mode e*Way Configuration

A Multi-Mode e*Way is a multi-threaded component used to route and transform data within e*Gate. Unlike traditional e*Ways, Multi-Mode e*Ways can use multiple simultaneous e*Way Connections to communicate with several external systems, as well as IQs or JMS IQ Managers. The following describes how to create and configure the Multi-Mode e*Way component for the LDAP e*Way. Multi-Mode e*Way properties are set using the Enterprise Manager.

3.1.1 Creating a Multi-Mode e*Way

1. Select the Navigator's Components tab.
2. Open the host on which you want to create the e*Way.
3. On the Palette, click on the **Create a New e*Way** button to create a new **e*Way**.
4. Enter the name of the new e*Way, then click **OK**.
5. Select the new e*Way component, right-click, and select **Properties**. The e*Way Properties dialog box opens.
6. The **Executable File** field defaults to **stceway.exe**. (stceway.exe is located in the "bin\" directory).
7. Type any additional command line arguments that the e*Way may require in the **Additional Command Line Arguments** field, taking care to insert them *at the end* of the existing command-line string. Be careful not to change any of the default arguments unless you have specific need to do so.
8. Click **New** under the **Configuration File** field to create a new configuration file, **Find** to select an existing configuration file, or **Edit** to edit the currently selected file. The Editor opens to edit settings for the Multi-Mode e*Way. The Multi-Mode e*Way Configuration Editor opens. The following section provides more information on these parameters.

9 After selecting the desired parameters, **Save** the configuration file and select **Promote to Run Time**. Click **OK** to close the e*Way Properties Window.

For more information on Multi-Mode e*Way settings and properties see the *e*Gate Integrator User's Guide, the Standard e*Way Intelligent Adapter User's Guide* or consult the e*Way Editor's online Help.

3.1.2 Multi-Mode e*Way Configuration Parameters

The Multi-Mode e*Way configuration parameters are organized in the following sections:

- **JVM Settings** on page 19
- **General Settings** on page 22

3.1.3 JVM Settings

The JVM Settings control basic Java Virtual Machine settings. The JVM Settings section contains the following parameters:

- **JNI DLL Absolute Pathname** on page 19
- **CLASSPATH Prepend** on page 20
- **CLASSPATH Override** on page 20
- **CLASSPATH Append From Environment Variable** on page 21
- **Initial Heap Size** on page 21
- **Maximum Heap Size** on page 21
- **Maximum Stack Size for Native Threads** on page 21
- **Maximum Stack Size for JVM Threads** on page 21
- **Disable JIT** on page 22
- **Remote debugging port number** on page 22
- **Suspend option for debugging** on page 22

JNI DLL Absolute Pathname

Description

Specifies the absolute pathname to where the JNI DLL installed by the *Java 2 SDK 1.3* is located on the Participating Host.

Required Values

A valid pathname.

Additional Information

The JNI dll name varies on different O/S platforms:

OS	Java 2 JNI DLL Name
Windows 2000/NT 4.0	jvm.dll
Solaris	libjvm.so
HP-UX	libjvm.sl

The value assigned can contain a reference to an environment variable, by enclosing the variable name within a pair of % symbols. For example:

```
%MY_JNIDLL%
```

Such variables can be used when multiple Participating Hosts are used on different platforms.

To ensure that the JNI DLL loads successfully, the Dynamic Load Library search path environment variable must be set appropriately to include all the directories under the Java 2 SDK (or JDK) installation directory that contain shared libraries (UNIX) or DLLs (NT).

CLASSPATH Prepend

Description

Specifies the paths to be prepended to the CLASSPATH environment variable for the Java VM.

Required Values

An absolute path or an environmental variable. This parameter is optional.

Additional Information

If left unset, no paths will be prepended to the CLASSPATH environment variable.

Existing environment variables may be referenced in this parameter by enclosing the variable name in a pair of % signs. For example:

```
%MY_PRECLASSPATH%
```

CLASSPATH Override

Description

Specifies the complete CLASSPATH variable to be used by the Java VM. This parameter is optional. If left unset, an appropriate CLASSPATH environment variable (consisting of required e*Gate components concatenated with the system version of CLASSPATH) will be set.

Note: *All necessary JAR and ZIP files needed by both e*Gate and the Java VM must be included. It is advised that the* ***CLASSPATH Prepend*** *parameter should be used.*

Required Values

An absolute path or an environmental variable. This parameter is optional.

Additional Information

Existing environment variables may be referenced in this parameter by enclosing the variable name in a pair of % signs. For example:

```
%MY_CLASSPATH%
```

CLASSPATH Append From Environment Variable

Description

Specifies whether the path is appended for the CLASSPATH environmental variable to jar and zip files needed by the Java VM.

Required Values

YES or **NO**. The configured default is YES.

Initial Heap Size

Description

Specifies the value for the initial heap size in bytes. If set to 0 (zero), the preferred value for the initial heap size of the Java VM will be used.

Required Values

An integer between 0 and 2147483647. This parameter is optional.

Maximum Heap Size

Description

Specifies the value of the maximum heap size in bytes. If set to 0 (zero), the preferred value for the maximum heap size of the Java VM will be used.

Required Values

An integer between 0 and 2147483647. This parameter is optional.

Maximum Stack Size for Native Threads

Description

Specifies the value of the maximum stack size in bytes for native threads. If set to 0 (zero), the default value will be used.

Required Values

An integer between 0 and 2147483647. This parameter is optional.

Maximum Stack Size for JVM Threads

Description

Specifies the value of the maximum stack size in bytes for JVM threads. If set to 0 (zero), the preferred value for the maximum heap size of the Java VM will be used.

Required Values

An integer between 0 and 2147483647. This parameter is optional.

Disable JIT

Description

Specifies whether the Just-In-Time (JIT) compiler will be disabled.

Required Values

YES or **NO**.

Note: *This parameter is not supported for Java Release 1.*

Remote debugging port number

Description

Specifies the port number for the remote debugging of the JVM.

Required Values

An integer between 2000 and 65536.

Suspend option for debugging

Description

Specifies whether the option for debugging will be enabled or suspended upon JVM startup.

Required Values

YES or **NO**.

3.1.4 General Settings

General Settings controls the period of time the workslice waits before it re-posts a message once a rollback has occurred. The General Settings section contains the following parameter:

- **Rollback Wait Interval** on page 22

Rollback Wait Interval

Description

Specifies the period of time in milliseconds, that the workslice waits before it re-posts a message once a rollback has occurred.

Required Values

An integer between 0 and 99999999.

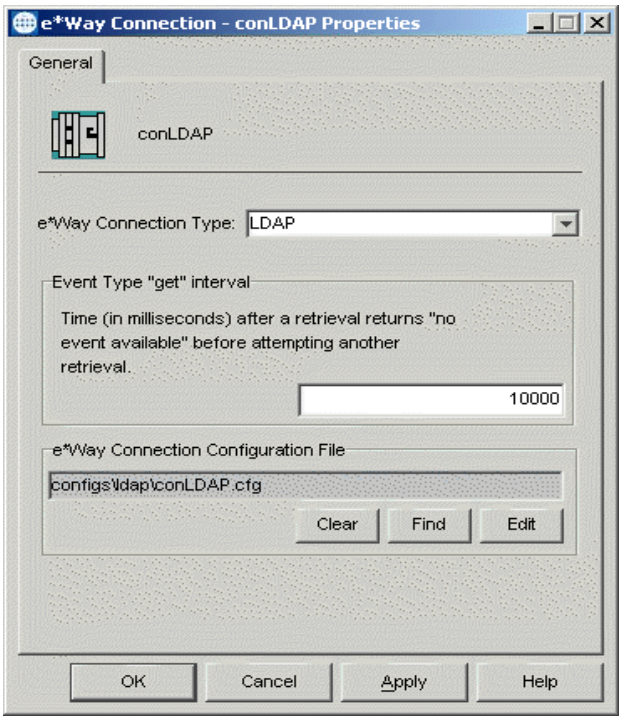
3.2 e*Way Connection Configuration

e*Way Connections are the encoding of access information for specific external connections. The e*Way Connection configuration file contains the parameters necessary for connecting with a specific external system. e*Way Connection parameters are set using the Enterprise Manager.

Creating an e*Way Connection

1. In the Enterprise Manager's Component editor, select the **e*Way Connections** folder.
2. On the palette, click the **Create a New e*Way Connection** button.
3. The **New e*Way Connection Component** dialog box opens. Enter a name for the new e*Way Connection and click **OK**.
4. Double-click on the new e*Way Connection. The **e*Way Connection Properties** dialog box opens.

Figure 3 e*Way Connection Properties



5. From the **e*Way Connection Type** drop-down box, select **LDAP**.
6. Enter the **Event Type "get"** interval in the dialog box provided. The configured default is 10000 milliseconds.
7. Click **New** under the **e*Way Connection Configuration File** field to create a new configuration file, **Find** to select an existing configuration file, or **Edit** to edit the currently selected file. The e*Way Connection Configuration Editor opens. The

following section provides more information on these e*Way Connection parameters.

8 After selecting the desired parameters, **Save** the configuration file and select **Promote to Run Time**. Click **OK** to close the e*Way Connection Properties Window.

Note: *If changes are made to an existing e*Way Connection file, any e*Ways using the revised e*Way Connection must be restarted.*

3.2.1 Configuring e*Way Connections

The LDAP e*Way Connection configuration parameters are organized into the following sections:

- **Connector** on page 24
- **Connection** on page 26
- **Referrals** on page 27

3.2.2 Connector

The LDAP Connector is associated with the **ldap.def** file. The **ldap.def** file is a template loaded by the e*Gate Enterprise Manager which enables the user to specify the LDAP connection properties when creating an instance of an LDAP Connector. The class implementing the LDAP Connector is:

com.stc.eways.jndi.runtime.NSConnector

The Connector defines the parameters for the LDAP connector class being used. It also defines the Connection Management properties for the Connection Manager facilities. This section contains a set of top level parameters:

- type
- class
- Property.Tag
- Connection Establishment Mode
- Connection Inactivity Timeout
- Connection Verification Interval

Important: *Do not change the default values for the following three parameters: type, class, and Property.Tag. These parameters relay information to the LDAP ETD about the LDAP Connector being used.*

Type

Description

Specifies the connector type.

Required Values

The default is **LDAP Connector** for Java LDAP connections.

Class

Description

Specifies the class name of the LDAP connector object.

Required Values

The default is **com.stc.eways.jndi.runtime.NSConnector**.

Property.Tag

Description

Specifies the data source identity. This parameter is required by the current EBobConnectorFactory.

Required Values

A valid data source package name.

Connection Establishment Mode

Description

Specifies how the connection with the LDAP server is established and closed.

- **Automatic** indicates that the connection is automatically established when the collaboration is started, and maintains the connection as needed.
- **OnDemand** indicates that the connection is established on demand as business rules requiring a connection to the external system are performed. The connection is closed once the methods are complete.
- **Manual** indicates that the user will explicitly call the connection open and close methods in the collaboration as business rules.

Required Values

Automatic, OnDemand or Manual. Automatic is the default.

Connection Inactivity Timeout

Description

Specifies timeout in milliseconds for the **Automatic** connection establishment mode. If it is not set, or set to zero, the continuous connection will not timeout due to inactivity. However if the connection goes down, it will automatically attempt to reestablish the connection. If a nonzero value is specified, the connection manager monitors for any inactivity and stops the connection if it reaches the specified value.

Required Values

An integer in the range of 0 to 864000.

Connection Verification Interval

Description

Specifies the minimum period of time in milliseconds between checks for connection status to the LDAP server. If the connection to the server is detected to be down during verification, the collaboration's onConnectionDown method is called. If the connection comes from a previous connection error, the collaboration's onConnectionUp method is called. If no value is specified, it defaults to 60000 milliseconds.

Required Values

An integer in the range of 0 to 864000.

3.2.3 Connection

This section contains a set of top level parameters:

- InitialContextFactory
- ProviderURL
- Authentication
- Principal
- Credentials

InitialContextFactory

Description

Creates the initial context to the LDAP server. The default value should not be changed.

Required Values

com.sun.jndi.ldap.LdapCtxFactory or other valid context factory

ProviderURL

Description

Specifies the URL connection string to create the initial context to the LDAP server.

For example, **ldap://ldap.seebeyond.com:389/dc=seebeyond,dc=com**.

The intial context is "**dc=seebeyond,dc=com"**.

Note: *The initial context is optional, depending on what context is established in the collaboration.*

Required Values

A valid URL text string.

Authentication

Description

Specifies the authentication mechanism required for connecting to the LDAP server.

- **none** specifies no authentication required to connect to the server.
- **simple** specifies simple username and password authentication for connecting to the server.

Required Values

Either **none** or **simple**.

Principal

Description

Specifies the principal (for example, username) when using an authentication mechanism other than anonymous login.

Required Values

A valid text string.

Credentials

Description

Specifies the credentials (for example, password) when using an authentication mechanism other than anonymous login.

Required Values

A valid encrypted text string.

3.2.4 Referrals

When the LDAP e*Way searches a directory for specific entries it may encounter a referral. Referrals are special entries in a directory that contain an LDAP URL referencing an actual entry on another LDAP directory. A referral can be automatically processed or followed by the e*Way, or it can be returned as a plain text entry. If the e*Way is configured to follow referrals, the e*Way will connect to the referred LDAP directory prior to executing the search on the referred LDAP directory.

Important: *Referrals are only valid for LDAP version 3 servers.*

This section allows the user to specify the behavior of the e*Way when it searches a directory for specific entries and encounters a referral. It contains a set of top level parameters:

- Follow
- CredentialsFile

Follow

Description

Specifies the behavior for LDAP version 3 referrals. If set to **No**, any referral entries in the directory will be ignored and returned as plain entries. If set to **Yes**, then the referral will be automatically followed.

Required Values

Yes or **No**. The configured default is **Yes**.

Note: *If the credentials specified in the **Connection** parameter are used for authentication when following the referrals, then do not specify the credentials file in the **CredentialsFile** parameter. If the referrals being followed require credentials that are different from the ones specified in the **Connection** parameter, then these credentials must be specified in the credentials file. The credentials file is specified using the **CredentialsFile** parameter. If set to Yes, and no credentials file is specified, then the LDAP e*Way will attempt to follow the referral using anonymous login.*

CredentialsFile

Description

Specifies the credentials file used when following any referrals in the directory. If this parameter is not set, then the credentials specified in the **Connection** section will be used for authentication when following the referrals. If this parameter is set with a credentials file, then the file will be used when following a referral.

Required Values

A valid full path to the credentials file.

Note: *If the credentials specified in the **Connection** section are used for authentication when following the referrals, then there is no need to specify the credentials file in the **CredentialsFile** parameter. If the referrals being followed require credentials that are different from the ones specified in the **Connection** section, then these credentials must be specified in the **Credentialsfile** parameter.*

The referral credentials file listed in the Help Tips for the **CredentialsFile** parameter can be created using the **com.stc.eways.jndi.RCFUtil** command-line interactive utility. See the following section **"Referrals Credentials File Utility (RCFUtil)"**.

3.2.5 Referrals Credentials File Utility (RCFUtil)

This section explains what to do for referrals that cannot be followed by specifying the Principal and Credential parameter settings. If the referrals can be followed using the Principal and Credentials specified in the **Connection** settings, then see **"Connection" on page 26**.

Referrals that require different credentials from the initial credentials used to create the initial connection, must be specified using a referrals credentials file. This file can be generated and maintained using a utility class called **com.stc.eways.jndi.RCFUtil**.

This utility is an interactive command-line program that stores the credentials in an encrypted format. To run this utility, set the Java classpath to the **stcnsclient.jar**, **stcjcs.jar**, and **gnu-getopt.jar** files. These JAR files are included in the e*Way installation and are located in **eGate/client/classes** and **eGate/client/ThirdParty/gnu-getopt/classes** respectively. See **"Files/Directories Created by the Installation" on page 16**.

Getting the Help Message

To display the help message describing the usage on **com.stc.eways.jndi.RCFUtil**, type:

```
java com.stc.eways.jndi.RCFUtil --help
```

The following message appears:

```
---+ RCFUtil (c) 2002 SeeBeyond +---

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 e*Way for following referrals
that required credentials different from those used to create the
connection to the initial LDAP server.

Usage : java com.stc.eways.jndi.RCFUtil OPTIONS -- <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.
```

The options are in GNU style and are specified using the long form (an option is prefixed with a double dash "--"). The credentials file is specified after the options with a terminating double dash at the end of the command line. When the credentials file is initially created, the utility requires a username and password. The username and password used to create the credentials file are required whenever the file is accessed. The options --username and --password can be used to specify the username and password at the command line. If the username and password are not specified with these options, then the utility will prompt for the username and password before proceeding. The --decrypt option can be used to tell the utility to display the Credentials as un-encrypted. This option can be used whenever the utility displays the Credentials.

Creating a Referrals Credentials File

Before beginning with a credentials file, it must be created using the **com.stc.eways.jndi.RCFUtil utility**.

The command for creating an RCF file and what displays is shown below:

```
java com.stc.eways.jndi.RCFUtil --create --username admin --
ldapRCF.rcf
Creating file ldapRCF.rcf...
Enter password >> seebeyond
File created!
```

The `--create` option is used to create a new referrals credentials file. The `--username` option is specified with "**admin**" as the username. The utility prompts for the password because the password was not specified with the `--password` option.

The name of the file created is **ldapRCF.rcf** under the current directory. The filename can be any valid filename on the OS platform and does not require any special conventions or extensions. A full path to the file can also be specified, for example, **C:\eGate\client\misc\ldapRCF.rcf on Windows**. Once created, entries can be added to the file and eventually used by the LDAP e*Way.

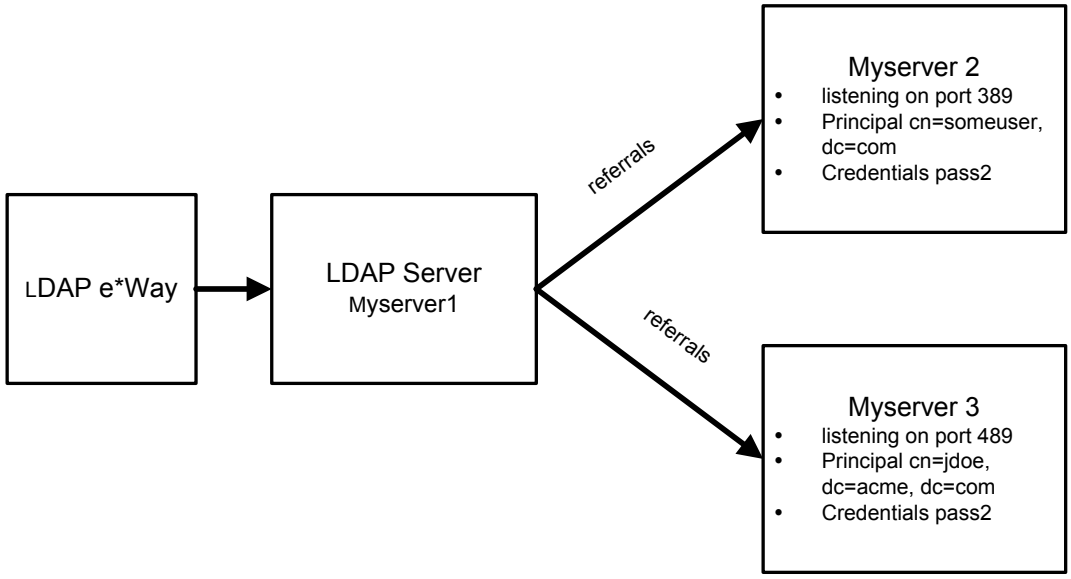
Adding a Credential to the Referrals Credentials File

When the LDAP e*Way encounters a referral and must authenticate the referred LDAP server, it looks up the referrals credentials file. The e*Way searches for an entry in the file with the matching host name and the port number it received from the referral. It then retrieves the Principal and Credentials from the matching entry and uses them to authenticate the referred LDAP server.

Note: *The user must know in advance which possible LDAP servers the e*Way may refer to as well as the required credentials for each LDAP server. The user cannot add an entry to the credentials file for each of the LDAP servers without this information.*

Figure 4 is an example of the structure of an LDAP server with referrals to other servers.

Figure 4 LDAP Server with Referrals



Using the information shown in Figure 4, the following section tells how to add entries to the credentials file.

1 To add myserver2 listening on port 389, issue the following command:

```
java com.stc.eways.jndi.RCFUtil --add -- ldapRCF.rcf
```

The utility then prompts for the following information:

- LDAP Host
- LDAP Port
- Principal
- Credentials.

```
Adding a referral credentials entry...
Enter username >> admin
Enter password >> seebeyond
Enter LDAP Host >> myserver2
Enter LDAP Port >> 389
Enter the Principal >> cn=someuser,dc=acme,dc=com
Enter the Credentials >> pass1

Done.
```

2 To add myserver3 listening on port 489, issue the following command:

```
java com.stc.eways.jndi.RCFUtil --add -- ldapRCF.rcf
```

Again, the utility then prompts for the following information:

- LDAP Host
- LDAP Port
- Principal
- Credentials.

```
Adding a referral credentials entry...
Enter username >> admin
Enter password >> seebeyond
Enter LDAP Host >> myserver3
Enter LDAP Port >> 489
Enter the Principal >> cn=jdoe,dc=acme,dc=com
Enter the Credentials >> pass2

Done.
```

Note: *To add an entry that uses anonymous login, do not enter the Principal information and press the <return> key.*

Displaying The Contents Of A Referrals Credentials File

The entries in the credentials file can be displayed by issuing the following command:

```
java com.stc.eways.jndi.RCFUtil --list -- ldapRCF.rcf
```

The entries are listed in no particular order. Each of the fields for an entry are separated by the pipe '|' character. The first field is the hostname, the second field is the port number, the third field is the Principal, and the fourth field is the encrypted Credentials.

```
Listing entries in the referral credentials file...
Enter username >> admin
Enter password >> seebeyond
1> myserver3 | 489 | cn=jdoe,dc=acme,dc=com | 05EDBA8021C7
2> myserver2 | 389 | cn=someuser,dc=acme,dc=com | 05EDBA80004D
```

To display the credentials in plain text, issue the following command:

```
java com.stc.eways.jndi.RCFUtil --list --decrypt -- ldapRCF.rcf
```

The `-decrypt` option tells the utility to display the encrypted credentials in plain text.

```
Listing entries in the referral credentials file...
Enter username >> admin
Enter password >> seebeyond
1> myserver3 | 489 | cn=jdoe,dc=acme,dc=com | pass2
2> myserver2 | 389 | cn=someuser,dc=acme,dc=com | pass1
```

Modifying An Entry In A Referrals Credentials File

The Principal and the Credentials entries are the only entries in the credentials file that can be modified with the RCFUtil utility. If the host and port number need to be changed, remove the existing entry matching that host and port number and then add an entry with the new host and/or port number. See **Removing An Entry From A Referrals Credentials File** on page 32 on how to remove an entry.

To modify either or both the Principal and Credentials information, issue the following command:

```
java com.stc.eways.jndi.RCFUtil --modify --username admin --password
seebeyond -decrypt
-- ldapRCF.rcf
```

The utility will display a list of entries in the credentials file. Select the entry number you want to modify. Then the utility will prompt for the new Principal and Credentials information. To change an entry to use anonymous login, do not enter the Principal and press <return>. The utility will ask for confirmation to change the entry to anonymous login.

```
Modifying entry from the referral credentials file...
NOTE : Only the Principal and Credentials can be modified.
       Use --remove to remove an entry.
1> myserver3 | 489 | cn=jdoe,dc=acme,dc=com | pass2
2> myserver2 | 389 | cn=someuser,dc=acme,dc=com | pass1
Enter number from list to modify>> 1
[myserver3:489] Principal >> cn=jdoe,dc=acme,dc=com
[myserver3:489] Enter new Principal >> cn=john doe,dc=acme,dc=com
[myserver3:489] Credentials >> pass2
[myserver3:489] Enter new Credentials >> pass3
Done.
```

Removing An Entry From A Referrals Credentials File

To remove an entry from the credentials file, issue the following command:

```
java com.stc.eways.jndi.RCFUtil --remove --username admin --password
seebeyond --decrypt -- ldapRCF.rcf
```

The utility will display a list of entries available in the credentials file. Select the number of the entry to remove.

```
Removing entry from the referral credentials file...
1> myserver3 | 489 | cn=john doe,dc=acme,dc=com | pass3
2> myserver2 | 389 | cn=someuser,dc=acme,dc=com | pass1
Enter number from list to remove>> 1
Done.
```

3.3 External Configuration Requirements

There are no configuration changes required in the external system. All necessary configuration changes can be made within e*Gate.

Chapter 4

LDAP ETD Overview

This chapter provides an overview of the LDAP ETD hierarchy structure, including the nodes, available methods and properties, and their application. For a more detailed description of each method see **"LDAP e*Way Classes and Methods" on page 89**.

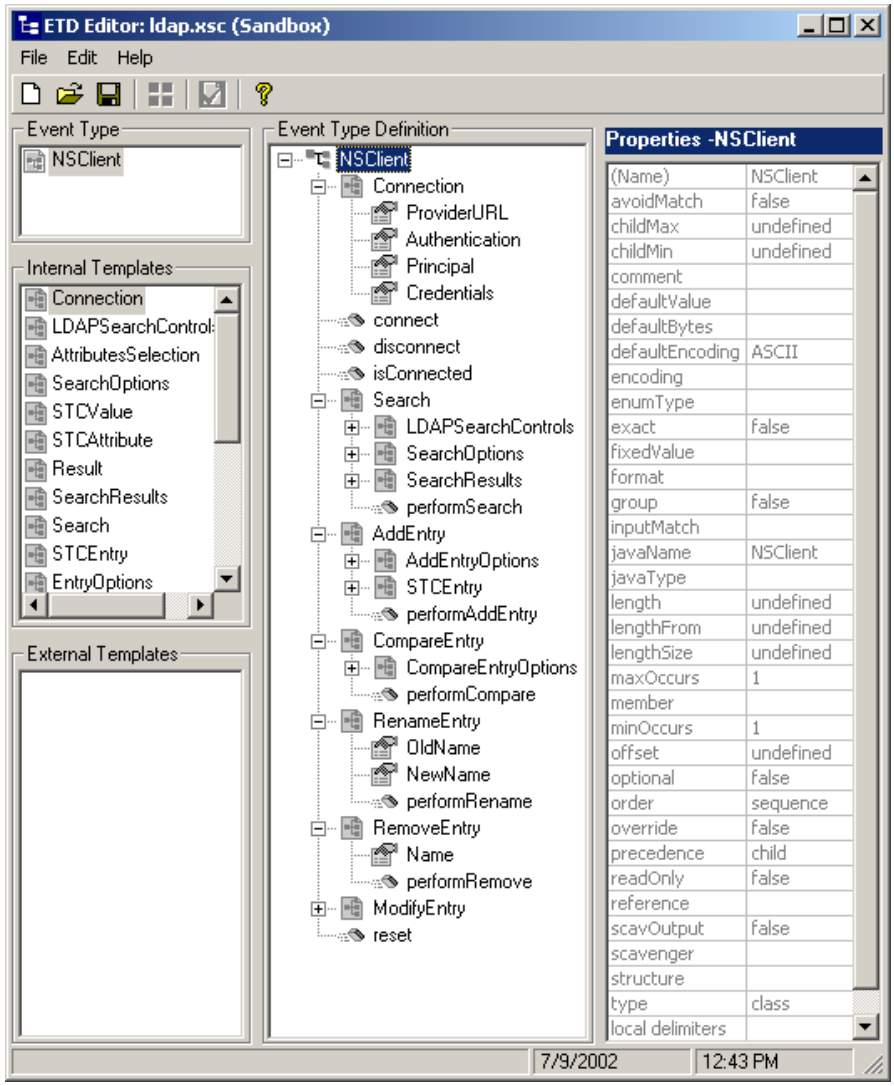
4.1 LDAP ETD Structure

The LDAP Event Type Definition (ETD) exposes the APIs for accessing an LDAP directory in the e*Gate Java collaboration environment. It is an uneditable read-only ETD. There are two components to the LDAP ETD: the **ldap.xsc** file, which exposes the structures and methods, and the Java classes, which implement those structures and methods. The following sections describe the LDAP ETD in detail and how to use the LDAP ETD in order to build a Java Collaboration rule for accessing an LDAP directory.

See next page

This section explains the structure and layout of the LDAP ETD (**ldap.xsc** file). Figure 5 shows an example of the LDAP ETD in the e*Gate Enterprise Manager ETD Editor Main window.

Figure 5 LDAP ETD in ETD Editor



The following is the general outline of the ETD and the methods and properties exposed on each node.

4.1.1 NSClient Root Node

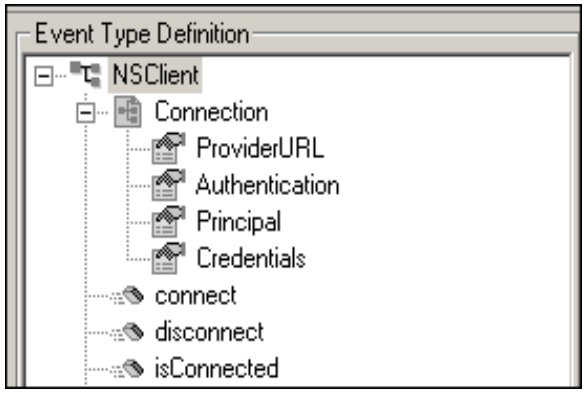
NSClient is the root node and provides a graphical representation of the interface. Expanding the node reveals all the methods and attributes on the interface, which are themselves represented as nodes. A node representing a method is normally expandable and reveals all the parameters for the method, as well as the return value (if present).

4.1.2 Connection Node

The Connection node populates the information required to connect to an LDAP server. This node specifies connection information when using the e*Way in Manual mode, where the connection to the LDAP server is handled manually by the user. The Connection node can only be used when the LDAP e*Way Connector is configured as "Manual" in the "Connection Establishment Mode" property of the "connector" properties. See **"Connector" on page 24** for details on this property.

Figure 6 shows the Connection node in its expanded form.

Figure 6 Connection Node in LDAP ETD



The Connection node has the following fields:

- ProviderURL
- Authentication
- Principal
- Credentials

The user must set the ProviderURL and the Authentication mechanism fields. Additionally, the user may be required to set the Principal and Credentials fields if the Authentication mechanism specified requires them. See **"Connection" on page 26** for details on the field properties.

Methods Under the Connection Node

Table 2 describes the exposed methods that manage connections to an LDAP server.

Table 2 Connection Node

Method Name	Description
connect	Creates a connection to the LDAP server using the specified parameters.
disconnect	Manually closes the connection to the LDAP server.
isConnected	Determines if the current connection to the LDAP server is still connected.

4.1.3 Search Node

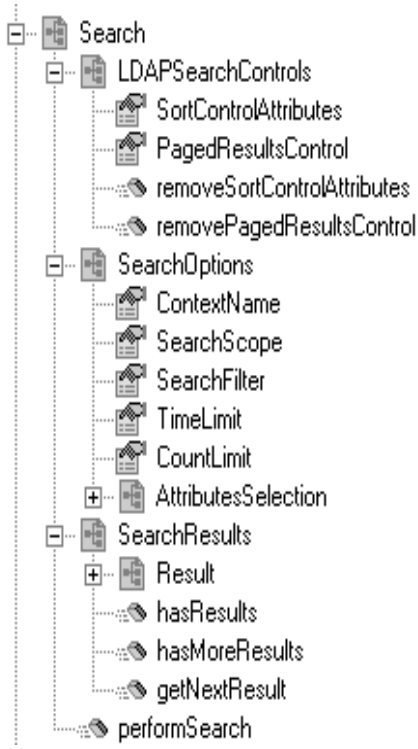
The Search node is specific to operations that are done once the e*Way is connected to the LDAP server. The Search node corresponds to performing searches for an entry or multiple entries of the LDAP directory.

To perform a search, the user specifies 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 leaf nodes, and fields are described in the following sections.

Figure 7 shows the Search node in its expanded form.

Figure 7 Search Node



The Search node has three nodes:

- LDAPSearchControls
- SearchOptions
- SearchResults,

and one method:

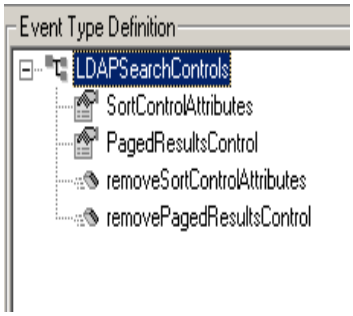
- performSearch.

To perform a search, the user first specifies any LDAP search controls to use, then specifies the search options such as the search filter, and then calls the performSearch method. Upon successfully returning from the performSearch method, the user can retrieve the results of the search by utilizing the SearchResults node.

LDAPSearchControls

Figure 8 shows the LDAPSearchControls node in its expanded form.

Figure 8 LDAPSearchControls Node



Important: *LDAP version 3 provides a way of extending functionality through the use of controls. Not all LDAP servers support controls or extensions. Before using controls, be sure to find out whether the LDAP server supports the controls being used. In addition, do not enable a particular control if the LDAP server does not support that particular control. Doing so will cause the e*Way to fail with an exception.*

Table 3 describes the fields exposed on the LDAPSearchControls Node.

Table 3 LDAPSearchControls Node

Field Name	Description
SortControlAttributes	Used to request that the results returned be sorted according to the attributes specified. To use sort control, set the SortControlAttributes 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 SortControlAttributes with a String that looks like `"cn|mail"`.
PagedResultsControl	Used to request that the results be returned in pages. Potentially, a search can result in hundreds or thousands of entries. If the server supports paged controls, then the user can set the PageResultsControl field with a number representing the number of entries returned per page of results.

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 will send the control information to the server. To remove the controls, use the removeSortControlAttributes or removePagedResultsControl methods. After a control is removed, subsequent searches will 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.

Figure 9 shows the SearchOptions node in its expanded form.

Figure 9 SearchOptions Node

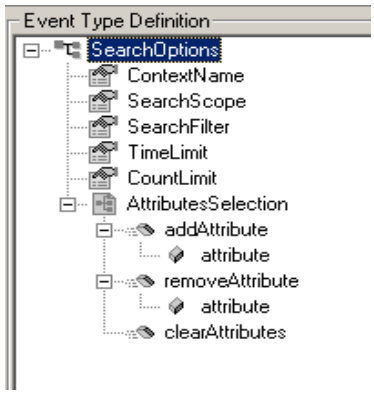


Table 4 describes the fields exposed on the LDAPSearchOptions Node.

Table 4 LDAPSearchOptions 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 ProviderURL. If the context name is not set correctly, then the e*Way will not be able to properly resolve the context name relative to the initial e*Way 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 Distinguished Name.
SearchScope	Used to set the scope or boundary of the search. When sending a search request, you must specify the scope of the search to identify the boundary of the search. This field is an integer type. For more information, see **"SearchOptions Scopes" on page 40**.
SearchFilter	Used to specify the search filter for the search, and is of type String. The basic search syntax is: `(attribute operator value)` ▪ 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 41** and **"SearchFilter Binary Operators" on page 41**. ***Note:*** *The search filter syntax is described by RFC 2254. For more information, refer to this RFC on www.ietf.org.*

Table 4 LDAPSearchOptions Node

Field Name	Description
TimeLimit	Used to specify the timeout in milliseconds for a search. If the search exceeds the set time limit, performSearch() will return without results.
CountLimit	Defines the maximum number of entries that can be returned on a search result.

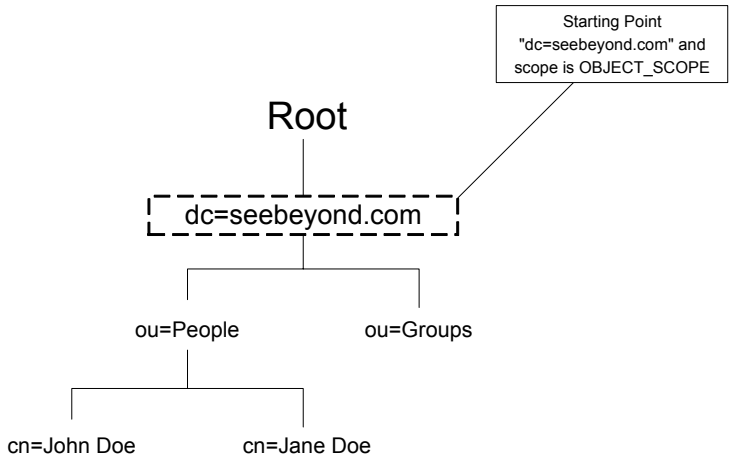
SearchOptions Scopes

The following section describes 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, type in the ETD field one of the following values described below as the scope parameter.

- **com.stc.eways.jndi.SearchOptions.OBJECT_SCOPE**

  SearchOptions.OBJECT_SCOPE tells the e*Way 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 10.

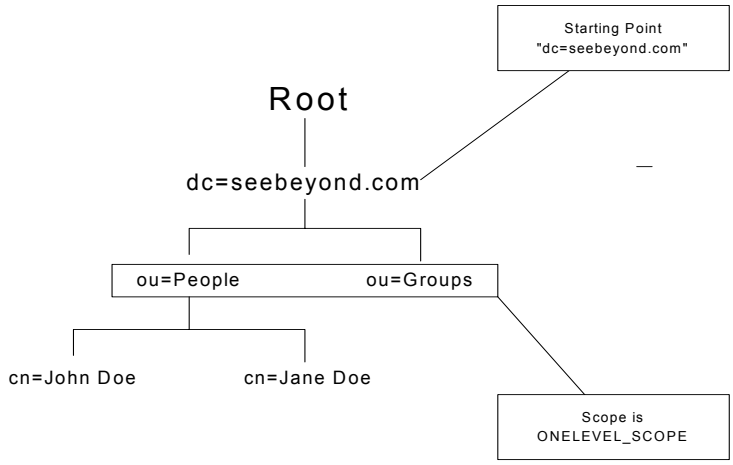
Figure 10 OBJECT_SCOPE



- **com.stc.eways.jndi.SearchOptions.ONELEVEL_SCOPE**

  SearchOptions.ONELEVEL_SCOPE tells the e*Way to search for entries one level below the named object. See Figure 11.

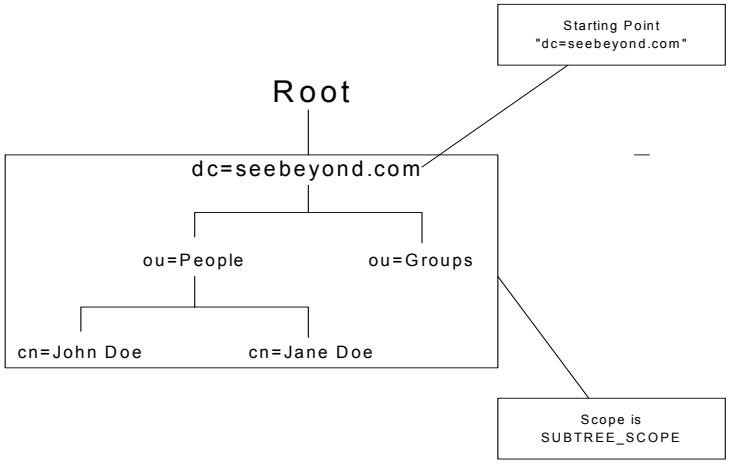
Figure 11 ONELEVEL_SCOPE



- **com.stc.eways.jndi.SearchOptions.SUBTREE_SCOPE**

  SearchOptions.SUBTREE_SCOPE tells the e*Way to search for all entries starting from the named object and all descendants below the named object. See Figure 12.

Figure 12 SUBTREE_SCOPE



SearchFilter Binary Operators

Additional operators that can be used in a filter expression are listed in Table 5.

Note: *Not all servers will support all the operators described in the LDAP e*Way User's Guide. See your LDAP server administrator on what search operators are supported.*

Table 5 Search Filter Binary Operators

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

SearchFilter Boolean Operators

Different conditions can be defined 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: &, |, !. For example, `(| (cn=John Doe)(sn=Smith))`, will get the entries with attribute "`cn`" equal to "`John Doe`" or entries with attribute "`sn`" equal to "`Smith`".

Note: *As already mentioned, the search filter syntax is described by RFC 2254. For more information, refer to this RFC on www.ietf.org.*

Boolean operators that can be used in a filter expression are listed in Table 6.

Table 6 Search Filter Boolean Operators

Operator	Comments	Example
&	Get all entries that match all the search filter criteria.	(& (sn=Smith)(telephoneNumber=444-4444)) will get entries with sn equal to Smith and telephoneNumber equal to 444-4444

Table 6 Search Filter Boolean Operators

Operator	Comments	Example
|	Get entries that match one or more of the search filter criteria.	(|(sn=Smith)(sn=Doe)) will get entries with sn equal to Smith or sn equal to Doe
!	Get entries that do not match the search filter criteria. Only one search filter can be specified (i.e., (! (filter)) is allowed but (!(filter)(filter)) is NOT allowed).	(! (sn=Smith)) gets all entries with attribute sn not equal to Smith

AttributesSelection

Under the SearchOptions node, the AttributesSelection node is used to restrict which attributes will be returned on a search. AttributesSelection is a collection of attribute IDs (names) that can be managed by using the **AddAttribute**, **RemoveAttribute**, and **ClearAttributes** 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 will remove any attributes added.

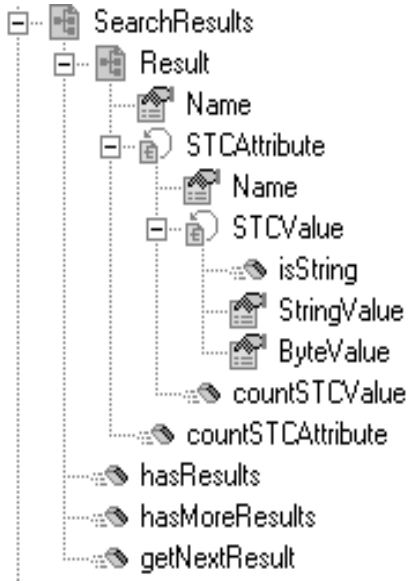
Important: *If **AddAttributesSelection** is not used, then all attributes are returned by default.*

SearchResults

The SearchResults node enables the user to retrieve the results returned by the search.

Figure 13 shows the SearchResults node in its expanded form.

Figure 13 SearchResults Node



The SearchResults node has the Result leaf node, and the following three methods:

- hasResults
- hasMoreResults
- getNextResult

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

To determine whether any 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 getNextResult() within a while loop.

The following example, taken from a Java collaboration rule, illustrates how to iterate through all the entries.

```
if (getLDAP().getSearch().getSearchResults().hasResults())
 {
    while (getLDAP().getSearch().getSearchResults().hasMoreResults())
    {
       getLDAP().getSearch().getSearchResults().getNextResult();
       System.out.println ("Entry>>>> " +
getLDAP().getSearch().getSearchResults().getResult().getName());
       System.out.println ("Attribute count : " +
getLDAP().getSearch().getSearchResults().getResult().countSTCAttribute());
       for( int i=0;i <
getLDAP().getSearch().getSearchResults().getResult().countSTCAttribute();i++)
       {
          System.out.println ("  Attribute : " +
getLDAP().getSearch().getSearchResults().getResult().getSTCAttribute(i).getName());
          System.out.println ("  Value count : " +
getLDAP().getSearch().getSearchResults().getResult().getSTCAttribute(i).countSTCValue());
          for( int j=0;j <
getLDAP().getSearch().getSearchResults().getResult().getSTCAttribute(i).countSTCValue();j++)
          {
             if
(getLDAP().getSearch().getSearchResults().getResult().getSTCAttribute(i).getSTCValue(j).isString()
)
```

```
                {
                   System.out.println ("    Value [String]: " +
getLDAP().getSearch().getSearchResults().getResult().getSTCAttribute(i).getSTCValue(j).getStringVa
lue());
                }
                else
                {
                   System.out.println ("    Value [byte array] : " +
StringUtil.toHexString(getLDAP().getSearch().getSearchResults().getResult().getSTCAttribute(i).get
STCValue(j).getByteValue()));
                }
             }
          }
       }
    }
```

The sample code listed above indicates the following:

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

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

```
System.out.println ("Entry>>>> " + getLDAP().getSearch().getSearchResults().getResult().getName());
```

Result

As already mentioned, calling getNextResult() populates Result with the next result. The Result node has the Name field, of type java.lang.String, which holds the Distinguished Name of the entry.

Result has the STCAttribute node which is a collection of attributes. To determine the number of STCAttribute in the collection, call the countSTCAttribute() method that returns an integer.

STCAttribute

Each STCAttribute has an attribute name, held in the Name field, and contains a collection of STCValue. To determine the number of STCValue in the collection, call the countSTCValue() method which returns an integer.

STCValue

Each STCValue represents the value of an attribute. A value can either be of type java.lang.String or a byte array (byte []). The user can determine the value type by calling the isString() method which will return true if the value is of type java.lang.String or false if the value is of type byte array (byte []). With the type determined, the user can retrieve the appropriate value using StringValue or ByteValue fields.

Retrieving Values for Attributes

Taking the example Java collaboration rule from the previous section, the following line of code shows how to retrieve the value for an attribute:

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

4.1.4 AddEntry

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 14 shows the AddEntry node in its expanded form.

Figure 14 AddEntry Node

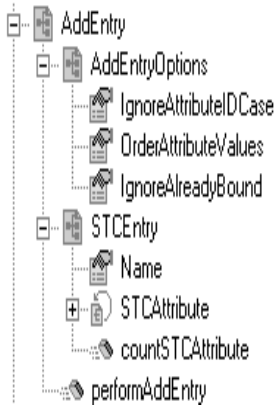


Table 7 describes the nodes and fields exposed on the AddEntry Node.

Table 7 AddEntry Node

Name	Description
AddEntryOptions node	Used to add entries to a directory and contains options when adding an entry.
▪ IgnoreAttributeIDCase field	Tells the e*Way 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 value for IgnoreAttributeIDCase is true.
▪ OrderAttributeValues field	Tells the e*Way 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 value for OrderAttributeValues is false.
▪ 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 e*Way to throw a CollabDataException when adding an already existed entry. An exception will be thrown if any other internal errors have occurred.
STCEntry node	Defines the entry to be added. The STCEntry node contains a collection of STCAttribute that can be used to define attributes for adding an entry.

Table 7 AddEntry Node

Name	Description
▪ Name field	Holds the name of the entry to be added and is of type java.lang.String. The name should be a RDN relative to the initial context created when the e*Way connected to the directory.
▪ STCAttribute	Each STCAttribute has a Name field of type java.lang.String that holds the name of the attribute. In addition, each STCAttribute contains a collection of STCValue.
▪ STCValue	Each STCValue holds either a java.lang.String or byte array (bye []) value. The correct type of value can be defined by setting either the StringValue or ByteValue field appropriately.

4.1.5 CompareEntry Node

The CompareEntry node is used to check for existing attribute(s) that have value(s). To compare an entry, the user specifies the RDN of the entry to compare and the "search" filter for the comparison. The user then invokes the performCompare() method that returns true if the specified entry has a matching attribute(s) with the values as specified in the filter.

Figure 15 shows the CompareEntry node in its expanded form.

Figure 15 CompareEntry Node

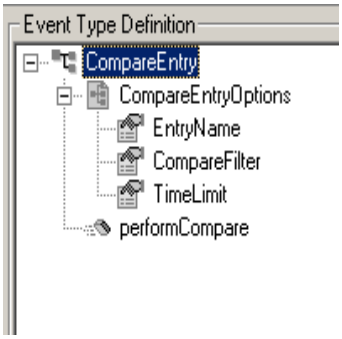


Table 8 describes the nodes and fields exposed on the CompareEntry Node.

Table 8 CompareEntry Node

Name	Description
CompareEntryOptions node	Used to check for existence of attribute(s) that have value(s).
▪ EntryName field	The DN of the entry to compare.

Table 8 CompareEntry Node

Name	Description
▪ 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() will return 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() will return false. An exception will be thrown if there were other internal errors. Example: comparing an entry that does not exist in the directory will result in a CollabDataException exception thrown because of a NameNotFoundException internal exception.
▪ TimeLimit field	Used to specify the timeout in milliseconds for a compare. If the compare exceeds the set time limit, performSearch() will return without results

4.1.6 RenameEntry Node

The RenameEntry node can be used 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() will fail with an exception.

Figure 16 shows the RenameEntry node in its expanded form.

Figure 16 RenameEntry Node

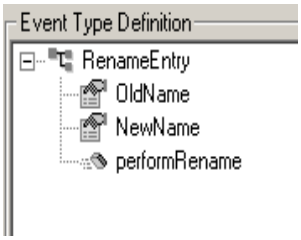


Table 9 describes the fields exposed on the RenameEntry Node.

Table 9 RenameEntry Node

Field Name	Description
OldName	The entry's existing name. Prior to calling the performRename() method, set the OldName field.
NewName	The entry's new name. Prior to calling the performRename() method, set the NewName field.

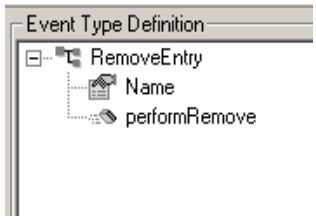
For both OldName and NewName fields, upon successfully renaming the entry, performRename() will return true; otherwise false is returned. An exception will be thrown if any other internal errors have occurred. For example, renaming an entry that does not exist in the directory will result in a CollabDataException exception thrown because of a NameNotFoundException internal exception.

4.1.7 RemoveEntry Node

The RemoveEntry node can be used to remove an entry from the directory. To remove an entry, the user specifies the RDN of the entry to remove and calls the performRemove() method.

Figure 17 shows the RemoveEntry node in its expanded form.

Figure 17 RemoveEntry Node



The RemoveEntry node has the Name field. The user can set the name of the entry to remove in the Name field. Once the name is set, call the performRemove() method to remove the specified entry from the directory. If the specified entry does not exist in the directory, then nothing occurs and there will be no exceptions thrown. An exception will be thrown if any other internal errors occur.

4.1.8 ModifyEntry Node

The ModifyEntry node is to modify an existing entry in the directory. The following modifications can be made to an entry:

1. Add attribute(s) to an entry.
2. Add value(s) to attribute(s) of an entry.
3. Remove attribute(s) from an entry.
4. Remove value(s) from attribute(s) of an entry.
5. Replace all existing value(s) of attribute(s) with new value(s) for an entry.

Modifications 1 and 2 are accomplished by using ModifyEntry's node called AddAttributesValues. Modifications 3 and 4 are accomplished by using ModifyEntry's node called RemoveAttributesValues. Modification 5 is accomplished by using ModifyEntry's node called ReplaceValues.

Figure 18 shows the ModifyEntry node in its expanded form.

Figure 18 ModifyEntry Node

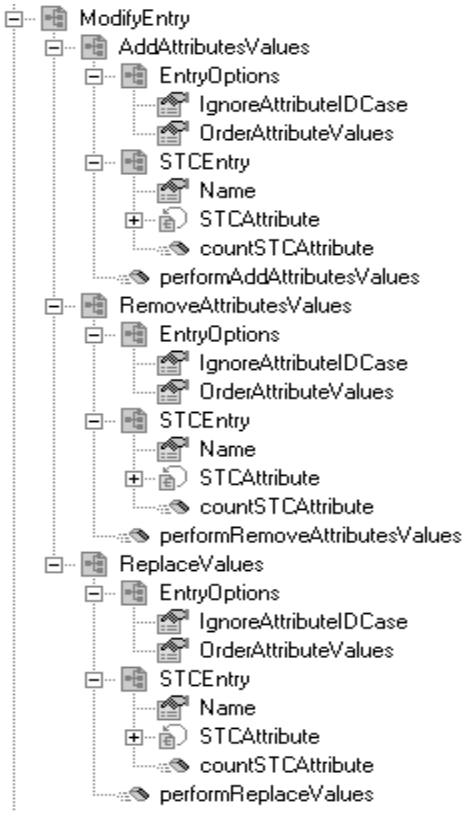


Table 10 describes the nodes and fields exposed on the ModifyEntry Node.

Table 10 ModifyEntry Node

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

Table 10 ModifyEntry Node

Name	Description
ReplaceValues node	Used to specify the entry to modify and the values for each of the attributes to replace. To replace the values of an attribute for an entry, specify the options, specify the name of the entry to modify, specify the attribute and values, and then call the performReplaceValues() method. If the modification is successful, then performReplaceValues () returns true. Otherwise, an exception is thrown or false will be returned. All existing values of an attribute are replaced by the newly specified values.
EntryOptions node	Used to specify options when **adding or removing** attributes and/or values to an entry. The following options can be set: ▪ IgnoreAttributeIDCase This field tells the e*Way 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 for IgnoreAttributeIDCase is true. ▪ OrderAttributeValues This field tells the e*Way 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 value for OrderAttributeValues is false.
STCEntry node	▪ The entry to modify, along with the attributes and/or values to **add**, **remove or replace**, can be specified using the STCEntry node. The STCEntry node contains the Name field. This field is of type java.lang.String and holds the name of the entry to be modified. The name should be a RDN relative to the initial context created when the e*Way connected to the directory. ▪ The STCEntry node contains a collection of STCAttribute that can be used to define attributes to be added or modified. ▪ Each STCAttribute has a Name field of type java.lang.String that holds the name of the attribute. ▪ In addition, each STCAttribute contains a collection of STCValue. Each STCValue holds either a java.lang.String or byte array (bye []) value. The correct type of value can be defined by setting either the StringValue or ByteValue field appropriately.
▪ performAddAttributesValues() method	The method to add attributes and/or values. See **"AddAttributesValues node"**.
▪ performRemoveAttributesValues() method	The method to remove attributes and/or values. See **"RemoveAttributesValues node"**.
▪ performReplaceValues()method	The method to replace values of an attribute. See **"ReplaceValues node"**.

4.1.9 LDAP ETD Java Classes

The LDAP .xsc structure, described in the previous sections, allows the user to perform drag and drop operations on the LDAP objects to create a Java Collaboration Rule. The code generated by the GUI will be compiled and requires the Java classes to implement the objects exposed in the **ldap.xsc** file.

The following table describes the Java classes that implement the LDAP ETD.

Table 11 LDAP ETD Java Classes

Class Name	Description
NSClient.java	Implements the ETDExt.java interface. It is the outer most class which holds an instance to each of the following classes: ▪ Search.java ▪ AddEntry.java ▪ CompareEntry.java ▪ ModifyEntry.java ▪ RemoveEntry.java ▪ RenameEntry.java ▪ ReplaceValues.java Each class corresponds to each of the nodes of the LDAP NSClient node in the **ldap.xsc** file.
Connection.java	Implements the LDAP connection properties set by the user to manually connect to the LDAP server.
Search.java	Implements the search functionality and holds an instance to each of the following classes: ▪ LDAPSearchControls.java ▪ SearchOptions.Java ▪ Result.Java
LDAPSearchControls.java	Implements the LDAP search controls to set and send to the LDAP server when invoking an LDAP search.
SearchOptions.java	Implements the search options to set when invoking an LDAP search. Options such as search filter, context name, and search scope are encapsulated with this class. SearchOptions.java holds an instance of an AttributesSelection.java class that implements the collection of desired attributes of a search result.
Result.java	Implements the LDAP search result object to use to iterate through the search results. Holds an instance of STCEntry.java class.
AddEntry.java	Implements the entry add functionality and holds an instance to each of the following classes: ▪ AddEntryOptions.java ▪ STCEntry.java.
AddEntryOptions.java	Implements the options set when adding an entry.
CompareEntry.java	Implements the entry compare functionality and holds an instance to CompareEntryOptions.java class. The CompareEntryOptions.java class implements the options to set when comparing an entry.
Rename.java	Implements the entry rename functionality.This class does not hold any references to any other classes.

Table 11 LDAP ETD Java Classes

Class Name	Description
Remove.java	Implements the entry removal functionality. This class does not hold any references to any other classes.
ModifyEntry.java	Implements the entry modification functionality and holds an instance to each of the following classes: ▪ AddAttributesValues.java ▪ RemoveAttributesValues.java ▪ ReplaceValues.java
AddAttributesValues.java	Implements the functionality for adding attributes to an entry or adding values to the attributes of an entry.
RemoveAttributesValues.java	Implements the functionality for removing attributes from an entry or removing values from the attributes of an entry. It holds an instance to each of the following classes: ▪ EntryOptions.java ▪ STCEntry.java
ReplaceValues.java	Implements the functionality for replacing existing values of attributes of an entry with new values. It holds an instance to each of the following classes: ▪ EntryOptions.java ▪ STCEntry.java
EntryOptions.java	Implements the options set prior to invoking the entry modification.
STCEntry.java	Implements the LDAP entry object. It contains a collection of attributes implemented by the STCAttributes.java class.
STCAttritubes.java	Holds instances of STCAttribute.java objects. Each STCAttribute.java object holds a collection of values implemented by the STCValues.java class.
STCValues.java	Holds instances of STCValue.java objects. Each STCValue.java object holds the actual value of either type java.lang.String or byte array (byte[]).

Note: *For more information on ETD nodes and methods, see the* ***e*Gate Integrator User's Guide****. For more information on Java classes and methods, see* **"LDAP e*Way Classes and Methods" on page 89***.*

Chapter 5

Implementation

This chapter presents information pertinent to implementing the Java-enabled LDAP e*Way in a production environment. The chapter provides an introduction to the e*Way components by demonstrating an implementation of the Search, Add, Modify, and Delete sample schemas.

The following assumptions apply to this implementation:

- The LDAP e*Way has been successfully installed.
- The executable and the configuration files have been appropriately assigned.
- All necessary .jar files are accessible.

5.1 LDAP e*Way Implementation

To complete the implementation of the LDAP e*Way, you will do the following:

- Make sure that the Control Broker is activated.
- In the e*Gate Enterprise Manager, define and configure the following as necessary:
  - Inbound and outbound e*Ways using **stcewfile.exe.**
  - The Multi-Mode e*Way component.
  - The e*Way Connection as described in **Chapter 3**.
  - Collaboration Rules to process Events.
  - Collaborations, to be associated with each e*Way component, to apply the required Collaboration Rules.
  - The destination to which data will be published prior to being sent to the external system.

The following sections describe how to define and associate each of the above components.

5.2 Sample Implementations

Five sample schemas for the LDAP e*Way are available in ..\samples\ewldap\.. on the installation CD-ROM. The sample implementations illustrate how to use e*Gate to search, add, delete, and modify the entries on an LDAP directory.

- The **"The Search Sample Schema" on page 57** demonstrates how the e*Way runs the business logic to search entries to an LDAP directory. The results of the search are then published to the JMS Queue, where they are picked up and published to a file.
- The Search sample schema with Referrals demonstrates how the e*Way runs the business logic to search entries to an LDAP directory with referrals. The results of the search are then published to the JMS Queue, where they are picked up and published to a file.

Note: *This sample has its own readme.txt file which explains how the sample schema is created. It is available in ..\samples\ewldap\.. on the installation CD-ROM.*

- The Add sample schema is described in **"Creating the Add, Modify, and Delete Sample Schemas" on page 79**. The Add sample demonstrates how the e*Way runs the business logic to add entries to an LDAP directory. The results of the addition are then published to the JMS Queue, where they are picked up and published to a file.
- The Modify sample schema **"Creating the Add, Modify, and Delete Sample Schemas" on page 79** demonstrates how the e*Way runs the business logic to modify entries to an LDAP directory. The results of the modification are then published to the JMS Queue, where they are picked up and published to a file.
- The Delete sample schema is described in **"Creating the Add, Modify, and Delete Sample Schemas" on page 79** demonstrates how the e*Way runs the business logic to delete entries to an LDAP directory. The results of the deletion are then published to the JMS Queue, where they are picked up and published to a file.

5.2.1 Importing the Sample Schema

The following pages describe how the components for the LDAP e*Way sample schema are created.

A **Readme.txt** is available in ..\setup\addons\ewldap\readme.txt on the installation CD-ROM. It provides the latest information on required ESRs and recent changes to the e*Way.

Note: *The Host and Control Broker are automatically created and configured during the e*Gate installation. The default name for each is the name of the host on which the e*Gate Enterprise Manager GUI is installed. For more information about creating or modifying any component within the e*Gate Enterprise Manager, see the e*Gate Enterprise Manager's online Help system.*

The first task in deploying the sample implementation is to create a new schema name. While it is possible to use the default schema for the sample implementation, it is recommended that you create a separate schema for testing purposes. After you install the LDAP e*Way, do the following:

1. Start the e*Gate Enterprise Manager GUI.
2. When the Enterprise Manager prompts you to log in, select the host that you specified during installation, and enter your password.
3. You will then be prompted to select a schema. Click **New**.
4. Enter a name for the new Schema. In this case, enter **LDAP_Sample**, or any name as desired.
5. To import the sample schema select **Create from Export**, and use **Find** to locate and select the sample .zip file on the CD-ROM. For Windows or UNIX operating systems select LDAP_Java.zip. Click **Open**.

Note: *If already running e*Gate, select **File**, **New Schema** to import the sample schema.*

The e*Gate Enterprise Manager opens under your new schema.

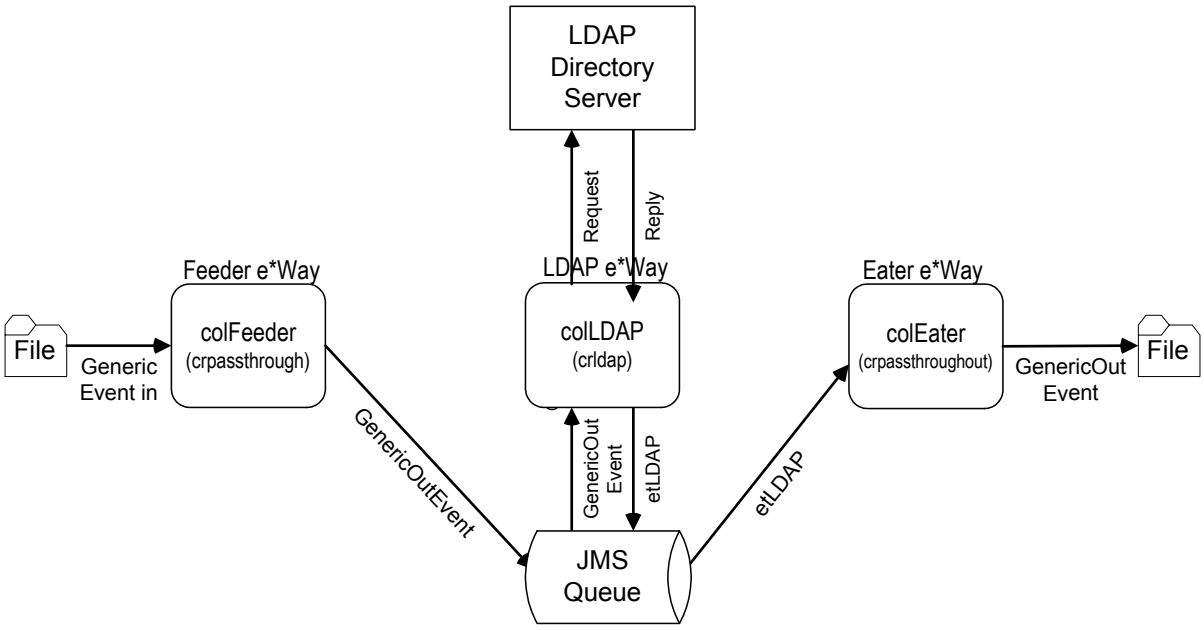
5.3 The Search Sample Schema

For the most part, all e*Way components are created when the sample schema is imported into e*Gate. This section describes in detail how the components are created and configured using the Search sample schema.

The Search sample demonstrates how the e*Way searches entries on an LDAP directory. The sample searches sample data entries and returns the results of the search in a file.

Figure 19 shows the flow of the e*Way: the inbound Feeder e*Way reads the sample file and publishes to the JMS Queue. The LDAP e*Way subscribes to the JMS Queue and sends a request to the LDAP directory. The LDAP directory returns the results and the LDAP e*Way publishes it to the JMS Queue. The Outbound Eater e*Way subscribes to the Queue and writes the results to file.

Figure 19 Schema Sample Component Architecture



Once the sample has been successfully imported into e*Gate, the user must configure it to correspond to the system as necessary. The following sections provide directions for creating the LDAP e*Way components used by the sample schema.

5.3.1 Creating and Configuring the e*Ways

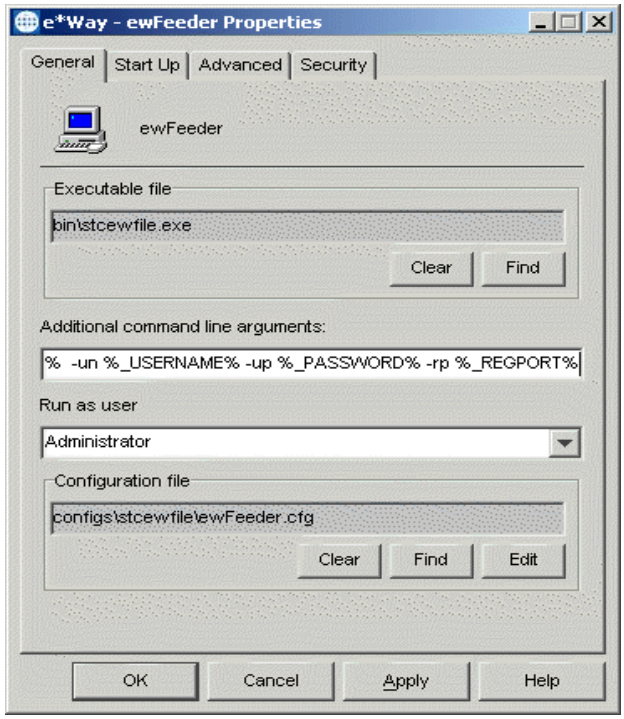
Component e*Ways connect with external systems to poll or send data. They also transform and route data. Multi-Mode e*Ways are used to run Java Collaborations that utilize e*Way Connections to send and receive Events to and from multiple external systems.

The Search sample schema contains three e*Ways, two of which are file-based (ewFeeder and ewEater) and one Multi-Mode (ewLDAP).

Creating the Inbound e*Way (ewFeeder)

1. Select the Navigator's **Components** tab.
2. Open the host on which you will create the e*Ways.
3. Select the **Control Broker** that will manage the new e*Ways.
4. On the palette, click the **Create a New e*Way** button.
5. Enter the name of the new e*Way (in this case "**ewFeeder**"), then click **OK**.
6. Right-click the new e*Way and select **Properties** to edit its properties.
7. The e*Way Properties window opens. Click the **Find** button beneath the **Executable File** field, and select **stcewfile.exe** as the executable file.

Figure 20 Inbound e*Way Properties dialog box



8 Under the **Configuration File** field, click the **New** (or Edit for saved e*Ways) button. The Configuration Editor opens. Select the following settings for this configuration file.

:

Table 12 Configuration Parameters for the Inbound e*Way

Parameter	Value
General Settings (unless otherwise stated, leave settings as default)	
AllowIncoming	YES
AllowOutgoing	NO
Outbound Settings	Default
Poller Inbound Settings	
PollDirectory	C:\INDATA (input file folder)
InputFileExtension	*.fin (input file extension)
PollMilliseconds	1000
Remove EOL	YES
MultipleRecordsPerFile	YES
MaxBytesPerLine	4096
BytesPerLineIsFixed	NO
File Records Per eGate Event	1
Performance Testing	Default

9 After selecting the desired parameters, save the **configuration** file (as "**ewFeeder**").

10 Click **File**, **Promote to Run Time**. This will close the Configuration Editor.

11 In the e*Way - Properties window, use the **Startup**, **Advanced**, and **Security** tabs to modify the default settings for each e*Way you configure.

   A Use the **Startup** tab to specify whether the e*Way starts automatically, or restarts after abnormal termination or due to scheduling, and so forth.

   B Use the **Advanced** tab to specify or view the activity and error logging levels, as well as the Event threshold information.

   C Use **Security** to view or set privilege assignments.

12 Select **OK** to close the e*Way Properties window.

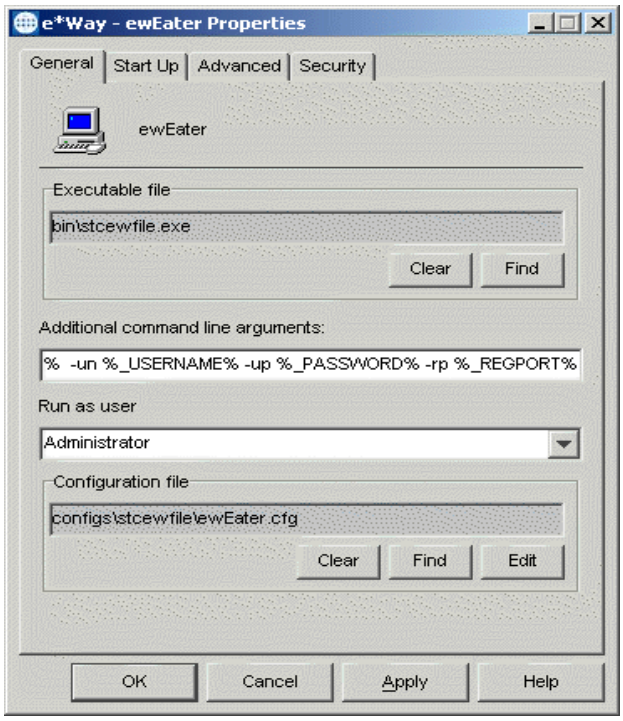
Creating the Outbound e*Way (Eater)

1 On the palette, click the **Create a New e*Way** button to create another e*Way.

2 Enter the name of the new e*Way (in this case "**ewEater**"), then click **OK**.

3 Select the new e*Way, right-click and select **Properties**. Click **Find** under the **Executable File** field and select **stcewfile.exe** as the executable file.

Figure 21 Outbound e*Way Properties dialog box



4 Under the **Configuration File** field, click the **New** button. The Configuration Editor opens. Select the following settings for this configuration file.

Table 13 Configuration Parameters for the Outbound e*Way

Parameter	Value
General Settings (unless otherwise stated, leave settings as default)	
AllowIncoming	NO
AllowOutgoing	YES
Outbound Settings	
OutputDirectory	C:\INDATA
OutputFileName	output%d.dat
MultipleRecordsPerFile	YES
MaxRecordsPerFile	10000
AddEOL	YES
Poller Inbound Settings	Default
Performance Testing	Default

5 Save the .cfg file as **ewEater**, and click **Promote to Run Time** to move the file to the runtime environment and close the Configuration Editor.

6 In the e*Way - Properties window, use the **Startup**, **Advanced**, and **Security** tabs to modify the default settings for the e*Way.

7 Click **OK** to close the e*Way Properties window.

Creating the Multi-Mode e*Way

1 Select the **Control Broker** that will manage the new e*Way.

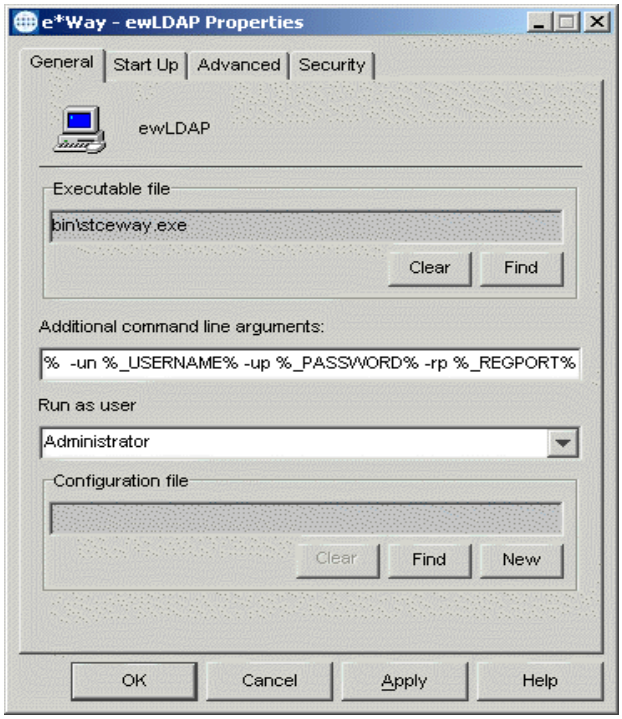
2 On the palette, click the **Create a New e*Way** button.

3 Enter the name of the new e*Way (in this case, "**ewLDAP**"), then click **OK**.

4 Right-click the new e*Way and select **Properties** to edit its properties.

5 When the e*Way Properties window opens, click the **Find** button beneath the **Executable File** field, and confirm that **stceway.exe** is the executable file (stceway.exe is the default).

Figure 22 Multi-Mode e*Way Properties dialog box.



6 To edit the JVM Settings, select **New** (or **Edit** if you are editing the existing .cfg file) under Configuration file. The Multi-Mode e*Way Configuration Editor opens.

See **Multi-Mode e*Way Configuration Parameters** on page 19 for details on the parameters associated with the Multi-Mode e*Way.

7 Save the **.cfg** file (**ewLDAP.cfg**), and **Promote to Run Time**.

8 In the e*Way Properties window, use the **Startup**, **Advanced**, and **Security** tabs to modify the default settings for each.

9 Click **OK** to close e*Way Properties window.

For more information on the Multi-Mode e*Way configuration settings see the *e*Gate Integrator User's Guide*.

5.3.2 Creating the e*Way Connection

An e*Way Connection is the encoding of access information for an external connection. The e*Way Connection configuration file contains the settings necessary for connecting with an LDAP server.

The **searchLDAPconn** e*Way Connection must be created for the Search sample. Use the following procedure to create the **searchLDAPconn** e*Way Connection.

1. Select the **e*Way Connection** folder on the **Components** tab of the e*Gate Navigator.
2. On the palette, click the **Create a New e*Way Connection** button.
3. Enter the name of the e*Way Connection (for this sample, "**searchLDAPconn**"), then click **OK.**
4. Double-click the new e*Way Connection to edit its properties.
5. The e*Way Connection Properties window opens. Select **LDAP** from the **e*Way Connection Type** drop-down menu.
6. Enter the **Event Type "get" interval** in the dialog box provided. 10000 milliseconds is the configured default. The "get" interval is the intervening period at which, when subscribed to, the e*Way connection is polled.
7. Under e*Way Connection Configuration File, click the **New** (Edit) button.
8. The e*Way Connection Editor opens. Select the following parameters listed in Table 14.

Important: *The LDAP e*Way enables the e*Gate system to contact the root directory identified by distinguished names. To search, modify, or perform other operations, the contact distinguished name must reside in a position above the subordinate node being accessed. Access to the various distinguished names is defined within the LDAP Server.*

9. For more information on the LDAP e*Way Connection parameters, see **e*Way Connection Configuration** on page 23.

Table 14 e*Way Connection Configuration Parameters

Parameter	Value
connector (unless otherwise stated, leave settings as default)	
type	LDAP Connector
class	com.stc.eways.jndi.runtime.NSConnector
Connection Establishment Mode	Automatic
Connection	
Initial Context Factory	com.sun.jndi.ldap.LdapCtxFactory
ProviderURL	ldap://eGate1.stc.com:12222/
Authentication	Simple
Principal	cn=root, o=STC, c=US

Table 14 e*Way Connection Configuration Parameters

Parameter	Value
Referrals	
Follow	NO

10 Save the .cfg file (**searchldapconn.cfg**) and **Promote to Run Time**.

Note: *The Provider URL parameter listed in Table 14 is used solely for the purpose of the Search sample. The Provider URL will vary based on the URL of the LDAP server.*

5.3.3 Event Types

The LDAP e*Way uses an Event Type Definition to parse, validate, and (if necessary) transform Events. For an overview of the LDAP ETD, see **Chapter 4**.

Each packet of data within e*Gate is referred to as an Event. An Event Type is a class of Events with a common data structure. The e*Gate system packages data within Events and categorizes them into Event Types. What these Events have in common defines the Event Type and comprises the ETD.

As described in **LDAP ETD Structure** on page 34, the **ldap.xsc** in not editable, and is a built-in component of the LDAP e*Way. However, additional custom ETDs must be created for the Search sample.

Creating an Event Types Using the Custom ETD Wizard

The following procedures show how to create an ETD (Event Type Definition) using the Custom ETD Wizard.

1 Highlight the **Event Types** folder on the **Components** tab of the e*Gate Navigator.

2 On the palette, click the **Create a New Event Type** button.

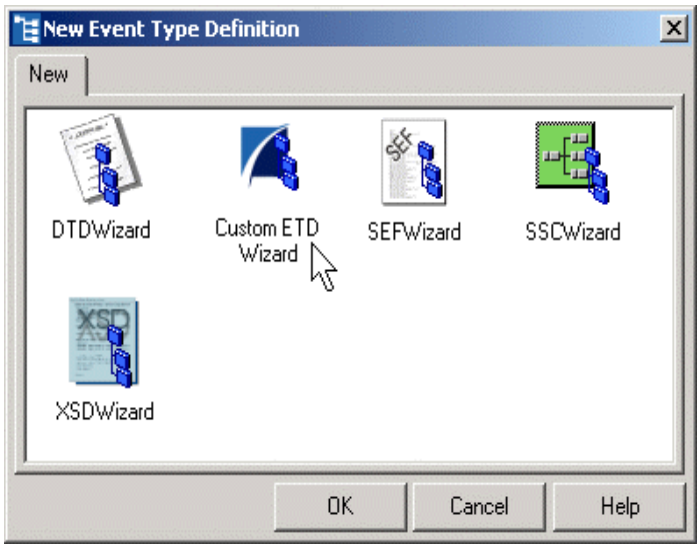
3 Enter the name of the **Event**, then click **OK.** For the purpose of this sample the Event Type is defined as **Blob**.

4 Double-click the new **Event Type** to edit its properties. The **Event Type Properties** dialog box opens.

5 Click the **New** button. The ETD Editor opens.

6 Select **New** from the File menu. The New Event Type Definition window opens.

Figure 23 Event Type Definition Wizards



7 Select the **Custom ETD** wizard.

8 Enter the Root Node Name (for this case, "**blob**").

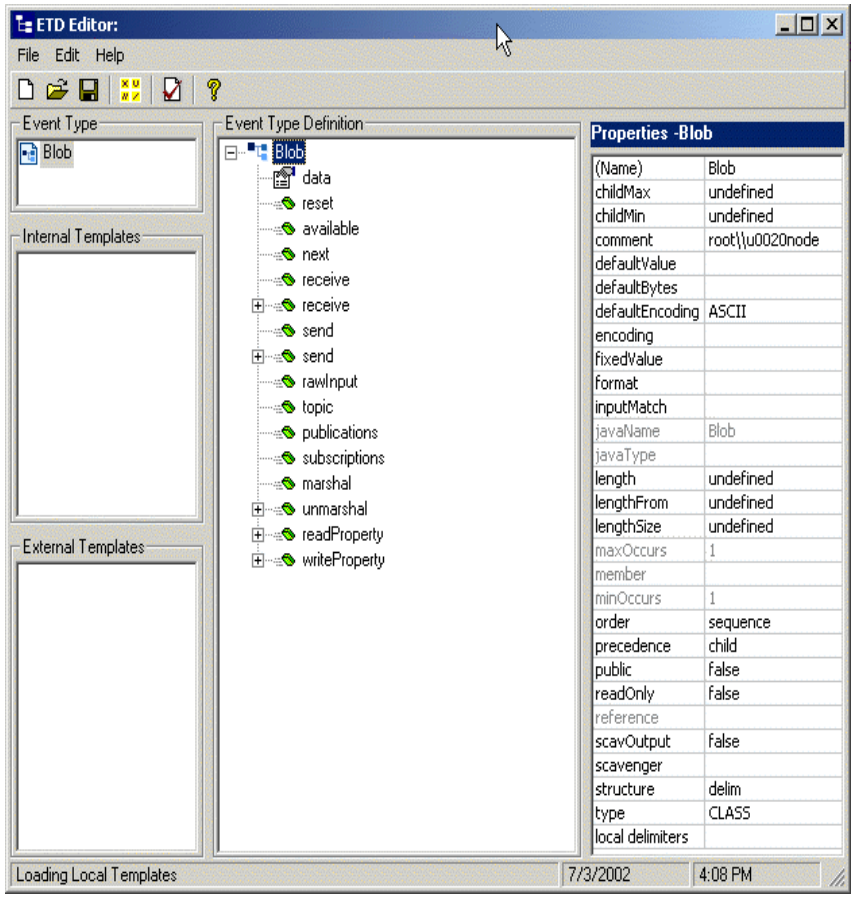
9 Enter a package name where the ETD Editor can place all the generated Java classes associated with the created ETD. (For this sample, use **Blob** as the package name.) Click **Next** and **Finish** to close the wizard.

10 Right click **Blob** in the Event Type Definition pane of the ETD Editor, and select **Add Field, as Child Node**.

11 Triple-click on **Field1**, and rename it **data**.

12 Select the **data** node. The properties for the data node are displayed in the Properties pane. Change the **structure** property to "**fixed**."

Figure 24 Event Type Definition Editor



13 From the File menu, click **Compile and Save**. Save the .xsc file as **blob.xsc**.

14 From the File menu, click **Promote to Run Time** to move the file to the run time environment.

15 Close the ETD Editor.

16 Click **OK** on the **Event Type Properties** dialog.

Creating Event Types and Associating Them with an Existing .xsc File

The following procedure shows how to create an Event Type Definition (ETD) from an existing .xsc file.

1 Select the **Event Types** folder on the **Components** tab of the e*Gate Navigator.

2 On the palette, click the **Create a New Event Type** button.

3 Type the name of the **Event Type** as **etLDAP** in the **New Event Type Component** window, then click **OK**.

4 Double-click the new **Event Type** to edit its properties. The **Event Type Properties** dialog box opens.

5 Click the **Find** button under the **Event Type Definition** field.

6 Browse to and select **ldap.xsc**.

7 Click **OK** to close the Event Type Properties dialog box.

5.3.4 Intelligent Queues

The next step is to create and associate **Intelligent Queues** (**IQs**). IQs manage the exchange of information between components within the e*Gate system and provide non-volatile storage for data as it passes from one component to another. IQs use IQ Services to transport data. IQ Services provide the mechanism for moving Events between IQs, and for handling the low-level implementation of data exchange (such as system calls to initialize or reorganize a database).

The Java implementation of the LDAP e*Way uses the SeeBeyond JMS (Java Message Service) IQ Service. For more information on the SeeBeyond JMS IQ see the ***SeeBeyond JMS Intelligent Queue User's Guide***.

Creating the Intelligent Queue

1 Select the Navigator's **Components** tab.

2 Open the host on which you want to create the IQ.

3 Open a **Control Broker**.

4 Select an IQ Manager.

5 On the palette, click the **Create a New IQ** button.

6 Enter the name of the new **IQ** (in this case "**iqJMS**"), then click **OK.**

7 Double-click the new **IQ** to edit its properties.

8 The Service defaults to **STC_JMS_IQ** and cannot be changed. This is true for any IQ created for an IQ Manager with the type **SeeBeyond_JMS**.

9 Set the initialization string and the Event Type Get Interval if necessary. The default Event Type Get Interval of 100 Milliseconds is satisfactory for the purposes of this implementation.

10 On the **Advanced** tab, make sure that **Simple publish/subscribe** is checked under the **IQ behavior** section.

11 Click **OK** to close the **IQ Properties** window

5.3.5 Creating Collaboration Rules

The next step is to create the Collaboration Rules that will extract and process selected information from the source Event Type defined above, according to its associated Collaboration Service. Creating Collaboration Rules includes assigning the subscription and publication instance name, Event Type, and specifying the e*Way Connection as either the source or the destination in the Collaboration. This binds the e*Way connection to the Collaboration Rule, enabling the collaboration to find the LDAP directory at runtime.

From the **Enterprise Manager Task Bar,** select **Options** and click **Default Editor**. The default should be set to **Java**.

Note: *Though the* ***Default Editor*** *can be set to either* ***Monk*** *or* ***Java,*** *the sample schemas require that the default be set to* ***Java****.*

The Search sample schema calls for the creation of three Collaboration Rules files.

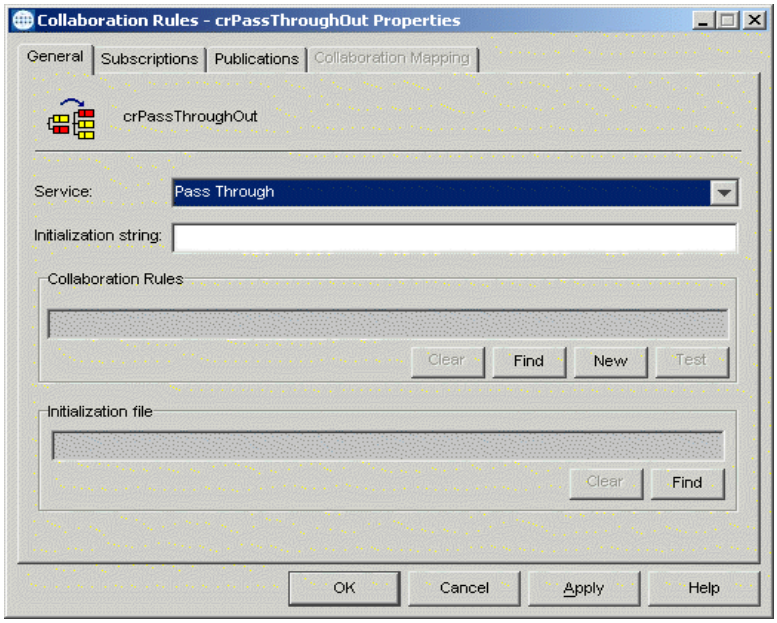
- **crPassThroughOut** (Pass Through)
- **crPassThroughIn** (Pass Through)
- **crLDAPsearch** (Java)

Collaboration Rules Files

crPassThroughOut (Pass Through)

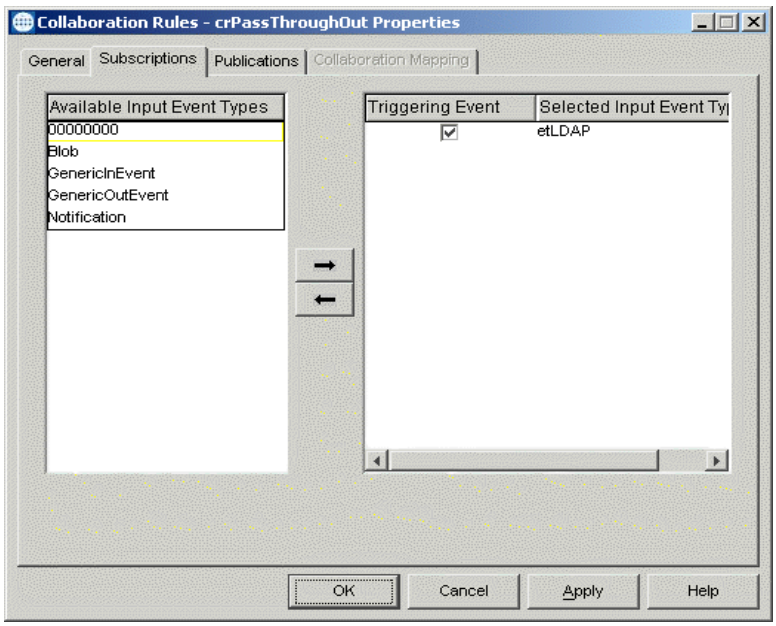
1. Select the Navigator's **Components** tab in the e*Gate Enterprise Manager.
2. In the Navigator, select the **Collaboration Rules** folder.
3. On the palette, click the **Create New Collaboration Rules** button.
4. Enter the name of the new Collaboration Rule Component (for this case "**crPassThroughOut**"), then click **OK**.
5. Double-click the new Collaboration Rules Component. The **Collaboration Rules Properties** window opens.

Figure 25 Collaboration Rules Properties - Pass ThroughOut



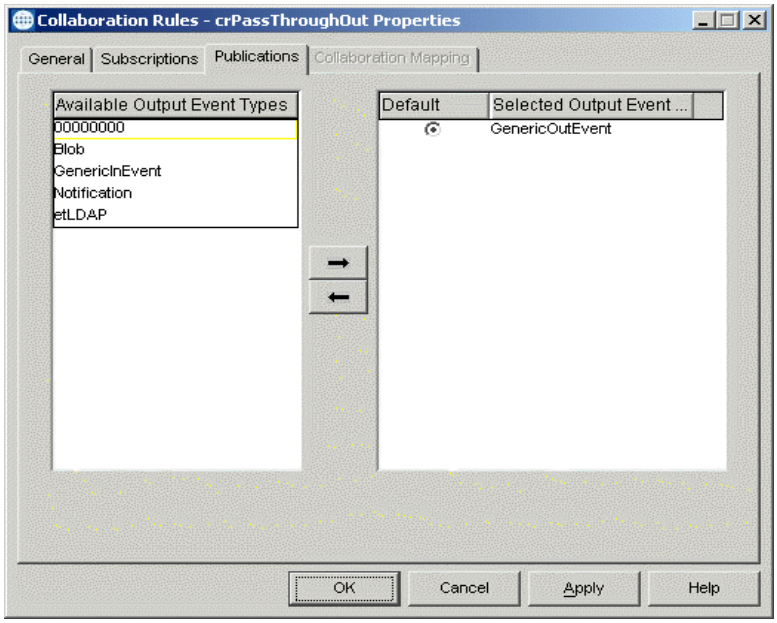
6. The **Service** field defaults to **Java**. From the **Service** field drop-down box, change to **Pass Through**.
7. Go to the **Subscriptions** tab. Select **etLDAP** under **Available Input Event Types**, and click the right arrow to move it to **Selected Input Event Types**. The box under **Triggering Event** should be checked. See Figure 26

Figure 26 Collaboration Properties--Subscriptions



8 Go to the **Publications** tab. Select **GenericOutEvent** under **Available Output Event Types**, and click the right arrow to move it to **Selected Output Event Types**. Make sure that **GenericOutEvent** is selected as the default. See Figure 27.

Figure 27 Collaboration Properties--Publications



9 Click **OK** to close the Collaboration Rules - crPassThroughout Properties window.

crPassThroughIn (Pass Through)

1 To create the **crPassThroughin Collaboration Rules** repeat steps 1–9, substituting:

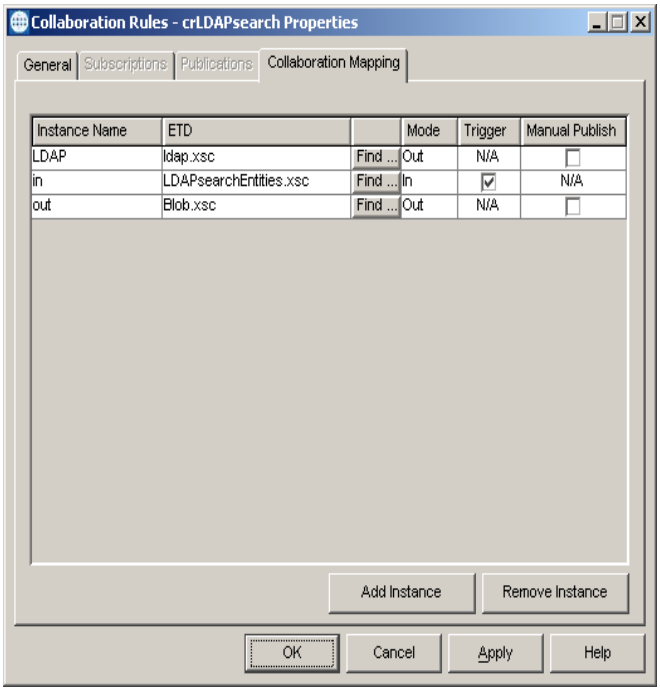
- **crPassThroughIn** for the name in step 4.

- **GenericInEvent** for the Selected Input Event in step 7.
- **GenericOutEvent** for the Selected Output Event in step 8.

crLDAPsearch (Java)

1. To create the Java Collaboration Rules select the **Collaboration Rules** folder.
2. On the palette, click the **Create New Collaboration Rules** button.
3. Enter the name of the new Collaboration Rule, then click **OK** (for this case, use **crLDAPsearch**).
4. Double-click the new Collaboration Rules Component to edit its properties. The Collaboration Rules Properties window opens.
5. The **Service** field defaults to **Java**.
6. In the **Initialization string** field, enter any required initialization string that the Collaboration Service may require. This field can be left blank.
7. Select the **Collaboration Mapping** tab.
8. Using the **Add Instance** button, create instances to coincide with the Event Types. For this sample, do the following.
9. In the Instance Name column, enter **LDAP** for the instance name.
10. Click **Find,** navigate to and double-click **ldap.xsc**. This adds **ldap.xsc** to the **ETD** column for this instance.
11. In the **Mode** column, select **Out** from the drop–down list box. To access the drop-down list box, click the right portion of the **Mode** field for this instance.
12. In the **Trigger** column, make sure that the checkbox is cleared (no trigger N/A).
13. In the **Manual Publish** column, make sure the checkbox is cleared (no Manual Publish).

Figure 28 Collaboration Rules - crLDAPSearch Properties Collaboration Mapping



14 Repeat steps 9–13 using the following values:

- Instance Name — **in**
- ETD — **Blob.xsc**
- Mode — **In**
- Trigger — selected
- Manual Publish - N/A

15 Repeat steps 9–13 again using the following values:

- Instance Name — **out**
- ETD — **Blob.xsc**
- Mode — **Out**
- Trigger — N/A
- Manual Publish - clear

The following section describes how the business logic for the Search sample schema is set up using the Java Collaboration Rules Editor.

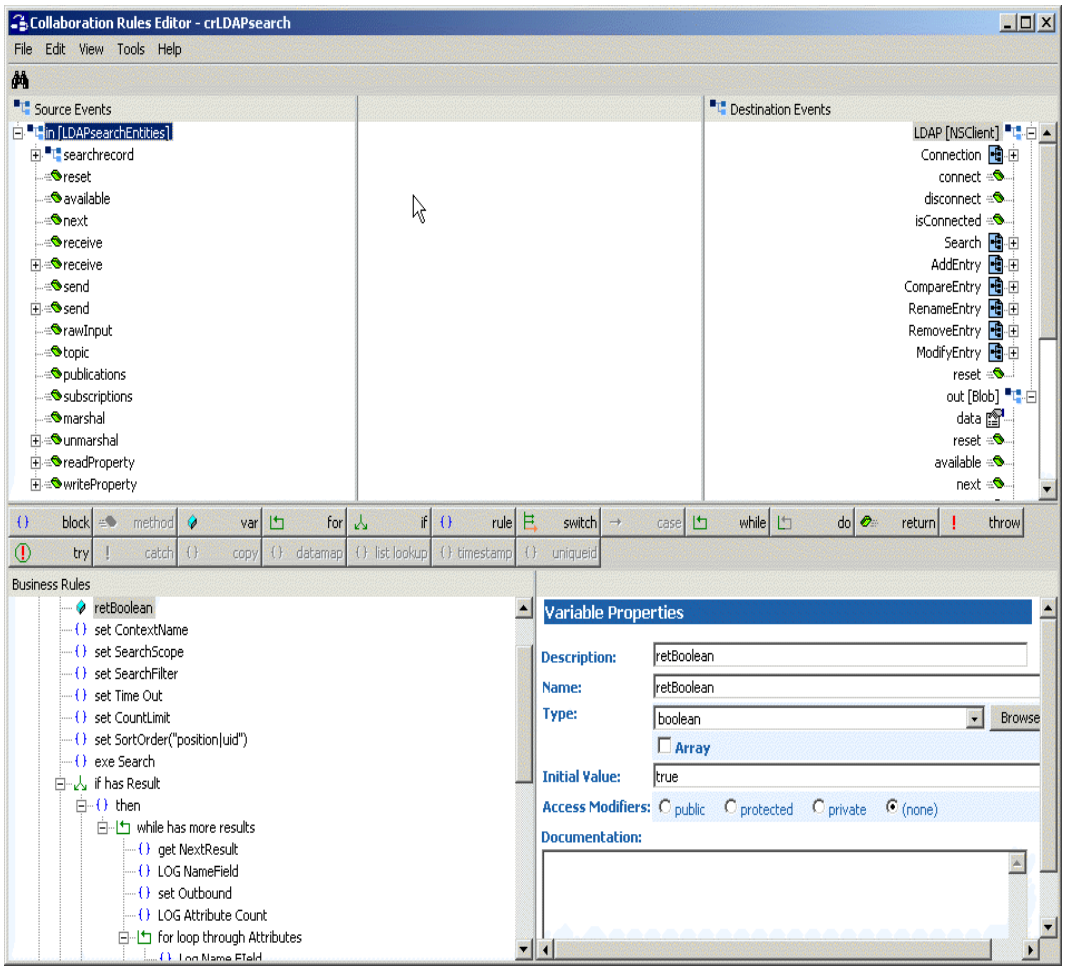
5.3.6 Creating Business Rules Using the Java Collaboration Rules Editor

The section provides an example of how to create business rules using the Java Collaboration Rules Editor. The completed Collaboration Rules .xpr file is included

with the Search sample schema on the installation CD-ROM. Refer to the completed class, **cr_ldap_search.class** when completing the Collaboration Rules Properties.

1 From the General tab, click the **New** or **Edit** button under the Collaboration Rules field. The Java Collaboration Rules Editor opens to the **cr_LDAP_search.xpr** from the Collaboration Rules Properties dialog box. Expand to full size for optimum viewing, expanding the Source and Destination Events as well.

Figure 29 The Collaboration Rules Editor



2 All of the user–defined business rules are added as part of execute business rules (). To view the created code in the Business Rules window, click **View** on the Menu bar and select **Display Code**.

Figure 30 Collaboration Rules Editor – Business Rules

Each new rule is created by clicking the **rule** button on the Business Rules toolbar, or by dragging and dropping one object to another from the Source and Destination Events panes. The following steps introduce how to create the **crLDAPsearch.xpr** business rules.

3 Click **Rule**. Drag **ContextName** located under **LDAP [NS Client]** (Destination Events) into the **Rule** pane. Type the value "**o=STC,c=US**" to create the following code:

```
getLDAP().getSearch().getSearchOptions().setContextName
("o=STC,c=US")
```

4 Click **Rule**. The **set SearchScope** rule is created by dragging **SearchScope** located under **LDAP [NS Client]** (Destination Events) into the **Rule** pane. Type the scope parameter to create the following code:

```
getLDAP().getSearch().getSearchOptions().setSearchScope(1)
```

Note: *See* **"SearchOptions Scopes" on page 40** *for more information on scope parameters.*

5 Click **Rule**. The **set SearchFilter** rule is created by dragging **SearchFilter** located under **LDAP [NS Client]** (Destination Events) into the **Rule** pane. Type the operator value to create the following code:

```
getLDAP().getSearch().getSearchOptions().setSearchFilter("name=*")
```

6 Click **Rule**. The **set Time Out** rule is created by dragging **TimeLimit** located under **LDAP [NS Client]** (Destination Events) into the **Rule** pane. Type the number value to create the following code:

```
getLDAP().getSearch().getSearchOptions().setTimeLimit(50000)
```

7 Click **Rule**. The **set CountLimit** rule is created by dragging **CountLimit** located under **LDAP [NS Client]** (Destination Events) into the **Rule** pane. Type the number value to create the following code:

```
getLDAP().getSearch().getSearchOptions().setCountLimit(100)
```

8 Click **Rule**. The **set SortOrder("position|uid")** rule is created by dragging **SortControlAttributes** located under **LDAP [NS Client]** (Destination Events) into the **Rule** pane. Type the value to create the following code:

```
getLDAP().getSearch().getLDAPSearchControls().setSortControlAttrib
utes("position|uid")
```

9 Click **Rule**. The **exe Search** rule is created by dragging **performSearch** located under **LDAP [NS Client]** (Destination Events) into the **Rule** pane.

```
getLDAP().getSearch().performSearch()
```

It then invokes the method.

Create the remaining business rules by using the business logic demonstrated in steps 1-9. You can also refer to the code of the **crldapsearch.xpr** business rules, available in ..\samples\ewldap\.. on the installation CD-ROM.

When all the business logic has been defined, compile the code by selecting **Compile** from the **File** menu. The **Save** menu opens. Provide a name for the **.**xpr file.

Note: *For more information on using the Java Collaboration Rules Editor, see the e*Gate Integrator User's Guide.*

5.3.7 Creating the Collaborations

Collaborations are the components that receive and process Event Types and forward the output to other e*Gate components or to an external. Collaborations consist of the Subscriber, which "listens" for Events of a known type (sometimes from a given source) and the Publisher, which distributes the transformed Event to a specified recipient.

Creating the colFeeder e*Way Collaboration

1 In the e*Gate Enterprise Manager, select the Navigator's **Components** tab.

2 Open the host on which you want to create the Collaboration, and select the **Control Broker.**

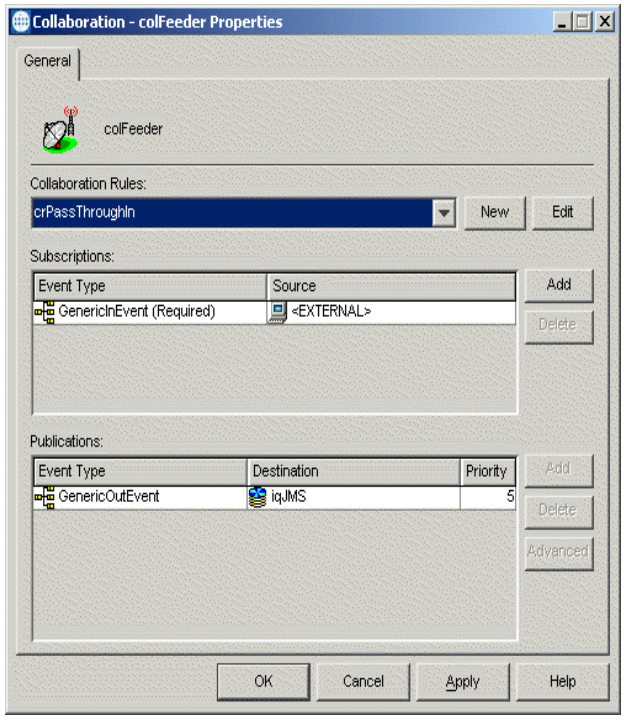
3 Select the **ewFeeder** e*Way to assign its Collaboration.

4 On the palette, click the **Create a New Collaboration** button.

5 Enter the name of the new Collaboration (for the sample, "**colFeeder**") then click **OK.**

6 Double-click the new Collaboration to edit its properties. The **Collaboration Properties** dialog box opens, as displayed in Figure 31.

Figure 31 Collaboration Properties colFeeder



7 From the **Collaboration Rules** drop-down list box, select the Collaboration Rules file that you previously created (for the sample, "**crPassThroughIn**").

8 In the **Subscriptions** area, click **Add** to define the input Event Type to which this Collaboration will subscribe.

A From the **Event Type** drop-down list box, select the **Event Type** "**GenericInEvent**."

B Select the **Source** from the **Source** drop-down list box. In this case, it should be **<External>**.

9 In the **Publications** area, click **Add** to define the output **Event Type** that this Collaboration will publish.

A From the **Event Type** list, select the **Event Type** "**GenericOutEvent**."

B Select the publication destination from the **Destination** list. In this case, it should be "**iqJMS**."

C The value in the **Priority** column defaults to **5**.

Creating the colLDAP e*Way Collaboration

1 In the e*Gate Enterprise Manager, select the Navigator's **Components** tab.

2 Open the host on which you want to create the Collaboration, and select the **Control Broker.**

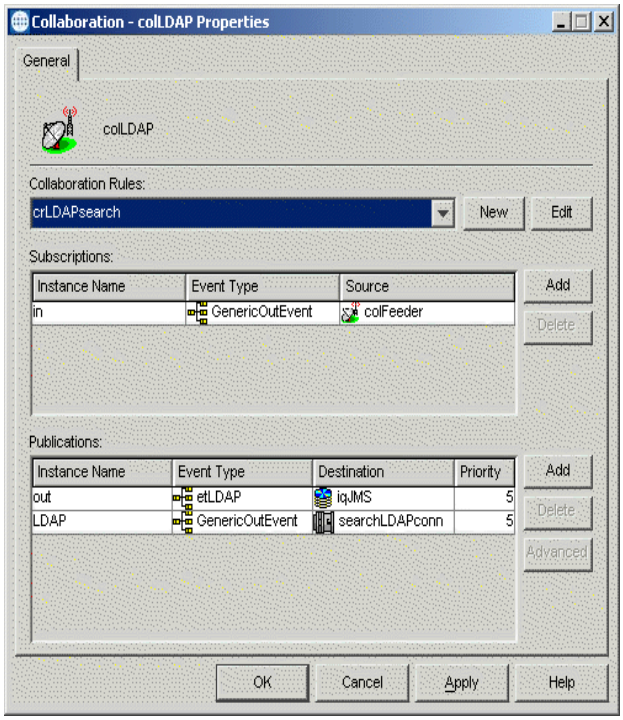
3 Select an e*Way to assign the Collaboration (for this sample, "**ewLDAP**").

4 On the palette, click the **Create a New Collaboration** button.

5 Enter the name of the new Collaboration, then click **OK.** (For the sample, "**colLDAP**")

6 Double-click the new **Collaboration** to edit its properties. The **Collaboration Properties** dialog box opens, as displayed in Figure 32.

Figure 32 Collaboration Properties colLDAP



7 From the **Collaboration Rules** drop-down list box select the Collaboration Rules file that you created previously (for the sample, "**crLDAPsearch**").

8 In the **Subscriptions** area, click **Add** to create an Instance Name to coincide with the Event Type. For this sample, type "**in**" for the Instance Name.

9 Define the input **Event Type** to which this Collaboration will subscribe.

A From the **Event Type** drop-down list box, select the **Event Type** "**GenericOutEvent**" provided with the sample schema.

B From the **Source** drop-down list box, select the source (for this sample "**colFeeder**").

10 In the **Publications** area, click **Add** to create an Instance Name to coincide with the Event Type. For this sample, type "**out**" for the Instance Name.

11 Define the output **Event Type** that this Collaboration will publish.

A From the **Event Type** drop-down list box, select the **Event Type** "**etLDAP**" provided with the sample schema.

B Select the publication destination from the **Destination** drop-down list box. In this case, it should be "**<iqJMS>**."

C The value in the **Priority** column defaults to **5**.

12 In the **Publications** area, click **Add** to create another Instance Name to coincide with the Event Type. For this sample, type "**LDAP**" for the Instance Name.

13 Define the output **Event Type** that this Collaboration will publish.

A From the **Event Type** drop-down list box, select the **Event Type** "**GenericOutEvent**" provided with the sample schema.

B Select the publication destination from the **Destination** drop-down list box. In this case, it should be "**<searchLDAPconn>**."

C The value in the **Priority** column defaults to **5**.

14 Click the **Apply** button and click **OK** to close the Collaboration Properties dialog box.

Creating the colEater e*Way Collaboration

1 In the e*Gate Enterprise Manager, select the Navigator's **Components** tab.

2 Open the host on which you want to create the Collaboration, and select the **Control Broker.**

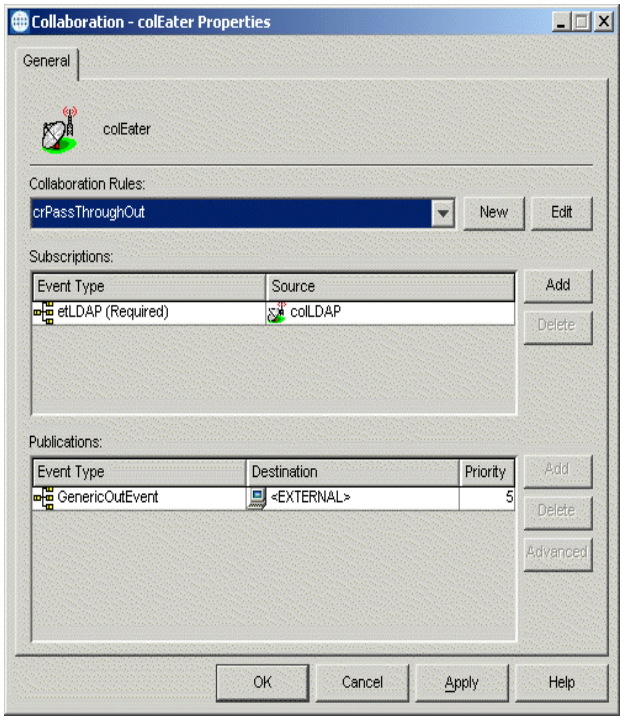
3 Select an e*Way to assign the Collaboration (for this sample, "**ewEater**").

4 On the palette, click the **Create a New Collaboration** button.

5 Enter the name of the new Collaboration, then click **OK.** (For the sample, "**colEater**")

6 Double-click the new **Collaboration** to edit its properties. The **Collaboration Properties** dialog box opens, as displayed in Figure 32.

Figure 33 Collaboration Properties colEater



7 From the **Collaboration Rules** drop-down list box select the Collaboration Rules file that you created previously (for the sample, "**crPassThroughOut**").

8 In the **Subscriptions** area, click **Add** to define the input Event Type to which this Collaboration will subscribe.

A From the **Event Type** drop-down list box, select the **Event Type** "**etLDAP**" provided with the sample schema.

B From the **Source** drop-down list box, select the source (for this sample "**colLDAP**").

9 In the **Publications** area, click **Add** to define the output **Event Type** that this Collaboration will publish.

A From the **Event Type** drop-down list box, select the **Event Type** "**GenericOutEvent**" provided with the sample schema.

B Select the publication destination from the **Destination** drop-down list box. In this case, it should be "**<External>**."

C The value in the **Priority** column defaults to **5**.

5.3.8 Executing the Schema

To execute the LDAP_Sample schema, do the following:

1 Go to the command line prompt, and enter the following:

```
stccb -rh hostname -rs LDAP_Sample -un username -up user password
-ln hostname_cb
```

Substitute ***hostname***, ***username*** and ***user password*** as appropriate.

2 Start the e*Gate Monitor GUI.

3 When prompted, specify the ***hostname*** which contains the Registry Host started in Step 1 above.

4 Select the LDAP_Sample schema.

5 After you verify that the Control Broker is connected (the message in the Control tab of the console will indicate command *succeeded* and status as *up*), select the IQ Manager, ***hostname***_igmgr, then right-click, and select **Start**.

6 Select each of the e*Ways, right-click, and select **Start**.

7 To view the output, copy the output file (specified in the Outbound e*Way configuration file). Save to a convenient location, open.

Note: *Opening the destination file while the schema is running will cause errors.*

5.4 Creating the Add, Modify, and Delete Sample Schemas

The Add, Modify, and Delete sample schemas contain e*Ways configured to add, modify, or delete the entries of an LDAP directory.

Note: *Modifying and deleting entries require that certain records and attributes be present on the LDAP server. Modifying and deleting entries should only be done once entries have been added to an LDAP directory, otherwise Java will throw an exception. The suggested running order is to first add, then modify or delete.*

The sections covering the Add, Delete, and Modify sample schemas only describe the aspects unique to those operations--the collaboration rules and the business rules.

However, these sample schemas have the same configuration parameters for the e*Ways and for the e*Way connections as the Search sample schema. For the e*Way configuration parameters, see **"Creating and Configuring the e*Ways" on page 58** and for the e*Way connection configuration parameters, see **"Creating the e*Way Connection" on page 63**.

For more detailed information on the steps required to create the sample schemas, refer to the **"The Search Sample Schema" on page 57**, which details each step involved in creating the Search sample schema. These same steps can be used to recreate each of the following sample schemas outlined in the following sections.

5.4.1 Creating the Add Sample Schema

The Add sample schema calls for the creation of three Collaboration Rules files.

- **crPassThroughOut** (Pass Through)

  See **"crPassThroughOut (Pass Through)" on page 68**.

- **crPassThroughIn** (Pass Through)

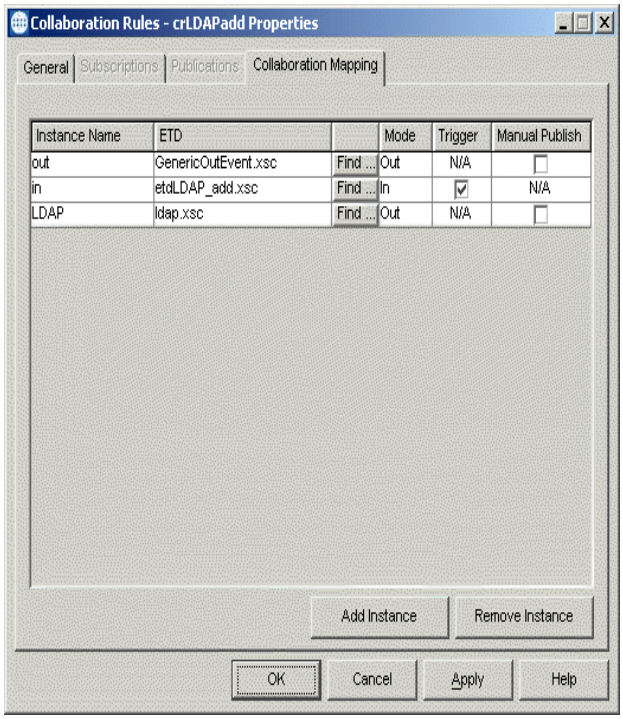
See **"crPassThroughIn (Pass Through)" on page 69**.

- **crLDAPadd** (Java)

For crLDAPadd, do the following:

1. To create the Java Collaboration Rules select the **Collaboration Rules** folder.
2. On the palette, click the **Create New Collaboration Rules** button.
3. Enter the name of the new Collaboration Rule, then click **OK** (for this case, use **crLDAPadd**).
4. Double-click the new Collaboration Rules Component to edit its properties. The Collaboration Rules Properties window opens.
5. From the **Service** field drop-down box, select **Java**. The **Collaboration Mapping** tab is now enabled, and the **Subscriptions** and **Publications** tabs are disabled.
6. In the **Initialization string** field, enter any required initialization string that the Collaboration Service may require. This field can be left blank.
7. Select the **Collaboration Mapping** tab.
8. Using the **Add Instance** button, create instances to coincide with the Event Types. For this sample, do the following.
9. In the Instance Name column, enter **LDAP** for the instance name.
10. Click **Find,** navigate to and double-click **ldap.xsc**. This adds **ldap.xsc** to the **ETD** column for this instance.
11. In the **Mode** column, select **Out** from the drop–down list box. To access the drop-down list box, click the right portion of the **Mode** field for this instance.
12. In the **Trigger** column, make sure that the checkbox is cleared (no trigger N/A).
13. In the **Manual Publish** column, make sure the checkbox is cleared (no Manual Publish).

Figure 34 Collaboration Rules - crLDAPadd Properties Collaboration Mapping



14 Repeat steps 9–13 using the following values:

- Instance Name — **in**
- ETD — **etdLDAPadd.xsc**
- Mode — **In**
- Trigger — selected
- Manual Publish - N/A

15 Repeat steps 9–13 again using the following values:

- Instance Name — **out**
- ETD — **GenericOutEvent.xsc**
- Mode — **Out**
- Trigger — N/A
- Manual Publish - clear

The following section describes how the business logic for the Add sample schema is set up using the Java Collaboration Rules Editor.

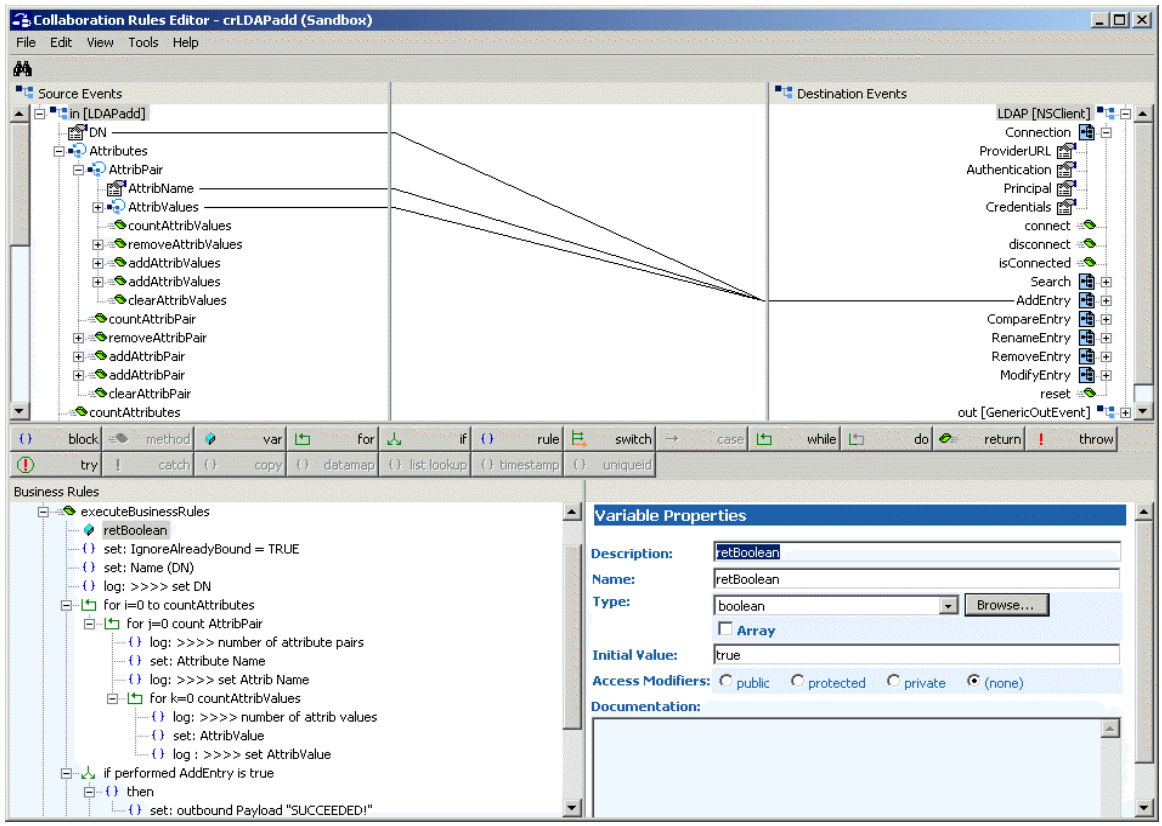
Creating Business Rules for the Add Sample Schema

The section provides an example of how to create the business rules for the Add sample schema using the Java Collaboration Rules Editor. The completed Collaboration Rules .xpr file is included with the Add sample schema on the installation CD-ROM.

Refer to the completed class, **crLDAPadd.class** when completing the Collaboration Rules Properties.

1 Click the **New** or **Edit** button under the Collaboration Rules field. The Java Collaboration Rules Editor opens to the **crLDAPadd.xpr** from the Collaboration Rules Properties dialog box. Expand to full size for optimum viewing, expanding the Source and Destination Events as well.

Figure 35 Collaboration Rules Editor crLDAPadd



Using the principles of the business logic detailed in the section **"Creating Business Rules Using the Java Collaboration Rules Editor" on page 71**, apply the business logic to create the crLDAPadd sample schema.

To create the Collaboration for Add sample schema, follow the steps in **"Creating the Collaborations" on page 74**. For step 8b, use "**colLDAPadd**" as the source type.

To execute the Add sample schema, see **"Executing the Schema" on page 78**.

5.4.2 Creating the Modify Sample Schema

The Modify sample schema calls for the creation of three Collaboration Rules files.

- **crPassThroughOut** (Pass Through)

  See **"crPassThroughOut (Pass Through)" on page 68**.

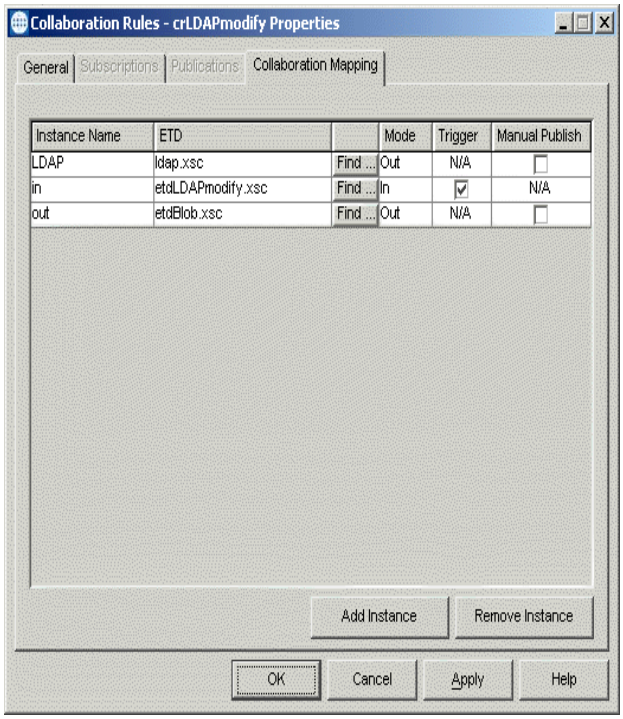
- **crPassThroughIn**(Pass Through)

See **"crPassThroughIn (Pass Through)" on page 69**.

- **crLDAPmodify** (Java)

1. To create the Java Collaboration Rules select the **Collaboration Rules** folder.
2. On the palette, click the **Create New Collaboration Rules** button.
3. Enter the name of the new Collaboration Rule, then click **OK** (for this case, use **crLDAPmodify**).
4. Double-click the new Collaboration Rules Component to edit its properties. The Collaboration Rules Properties window opens.
5. From the **Service** field drop-down box, select **Java**. The **Collaboration Mapping** tab is now enabled, and the **Subscriptions** and **Publications** tabs are disabled.
6. In the **Initialization string** field, enter any required initialization string that the Collaboration Service may require. This field can be left blank.
7. Select the **Collaboration Mapping** tab.
8. Using the **Add Instance** button, create instances to coincide with the Event Types. For this sample, do the following.
9. In the Instance Name column, enter **LDAP** for the instance name.
10. Click **Find,** navigate to and double-click **ldap.xsc**. This adds **ldap.xsc** to the **ETD** column for this instance.
11. In the **Mode** column, select **Out** from the drop–down list box. To access the drop-down list box, click the right portion of the **Mode** field for this instance.
12. In the **Trigger** column, make sure that the checkbox is cleared (no trigger N/A).
13. In the **Manual Publish** column, make sure the checkbox is cleared (no Manual Publish).

Figure 36 Collaboration Rules - crLDAPmodify Properties Collaboration Mapping



14 Repeat steps 9–13 using the following values:

- Instance Name — **in**
- ETD — **etdLDAPmodify.xsc**
- Mode — **In**
- Trigger — selected
- Manual Publish - N/A

15 Repeat steps 9–13 again using the following values:

- Instance Name — **out**
- ETD — **etdBlob.xsc**
- Mode — **Out**
- Trigger — N/A
- Manual Publish - clear

The following section describes how the business logic for the Modify sample schema is set up using the Java Collaboration Rules Editor.

Creating Business Rules for the Modify Sample Schema

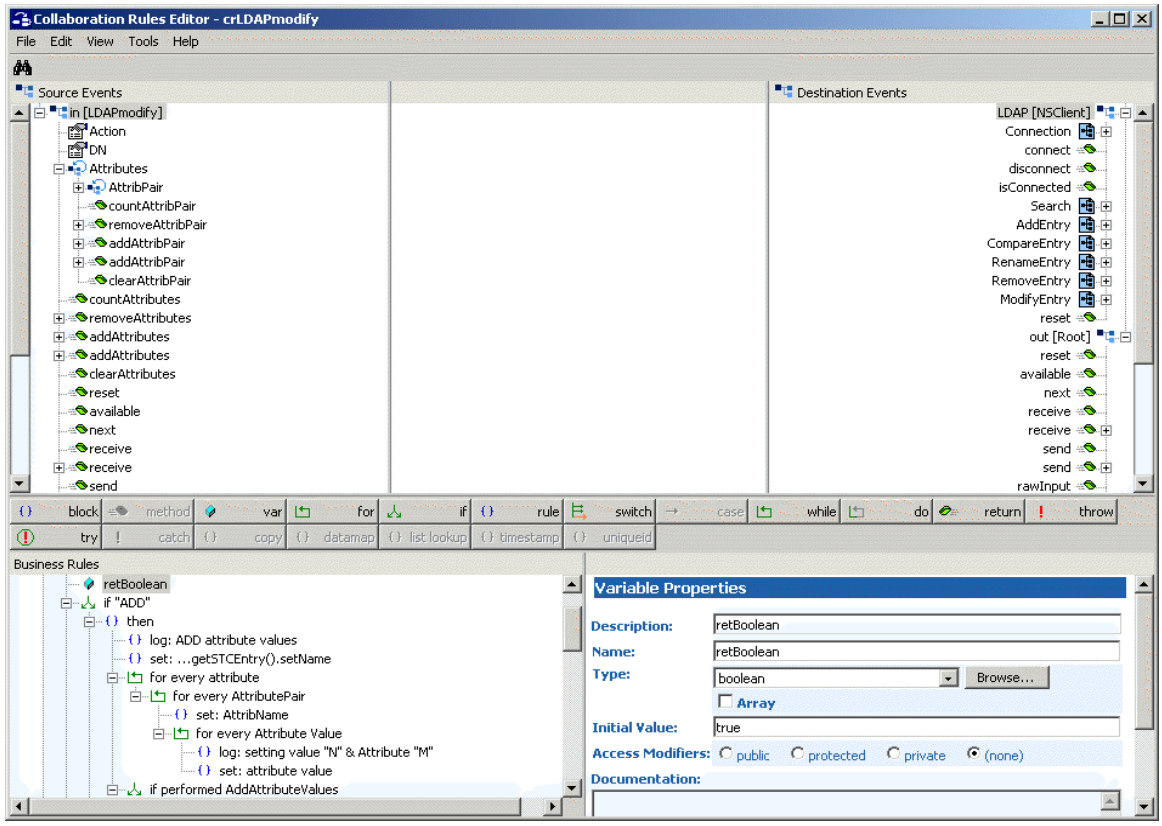
The section provides an example of how to create the business rules for the Modify sample schema using the Java Collaboration Rules Editor. The completed Collaboration Rules .xpr file is included with the Modify sample schema on the installation CD-ROM.

Note: *While modifying entries, be careful not to add an existing value. This will result in an exception caused by an Attribute In Use Exception exception. See* **"ModifyEntry Node" on page 49** *for details.*

Refer to the completed class, **crLDAPmodify.class** when completing the Collaboration Rules Properties.

1 Click the **New** or **Edit** button under the Collaboration Rules field. The Java Collaboration Rules Editor opens to the **crLDAPmodify.xpr** from the Collaboration Rules Properties dialog box. Expand to full size for optimum viewing, expanding the Source and Destination Events as well.

Figure 37 Collaboration Rules Editor crLDAPmodify



Using the principals of the business logic detailed in the section **"Creating Business Rules Using the Java Collaboration Rules Editor" on page 71**, apply the business logic to create the crLDAPmodify sample schema.

To create the Collaboration for Modify sample schema, follow the steps in **"Creating the Collaborations" on page 74**. For step 8b, use "**colLDAPmodify**" as the source type.

To execute the Modify sample schema, see **"Executing the Schema" on page 78**.

5.4.3 Creating the Delete Sample Schema

The Delete sample schema calls for the creation of three Collaboration Rules files.

- **crPassThroughOut** (Pass Through)

  See **"crPassThroughOut (Pass Through)" on page 68**.

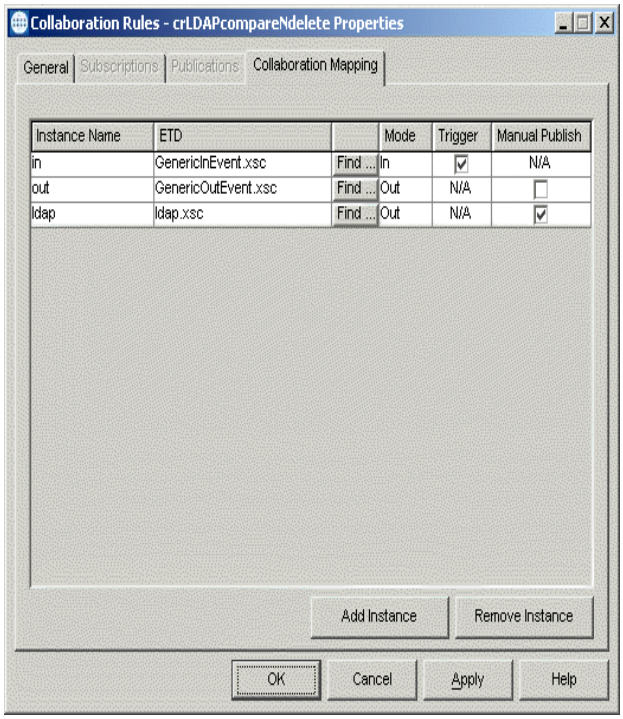
- **crPassThroughIn**(Pass Through)

  See **"crPassThroughIn (Pass Through)" on page 69**.

- **crLDAPcompareNdelete** (Java)

1. To create the Java Collaboration Rules select the **Collaboration Rules** folder.
2. On the palette, click the **Create New Collaboration Rules** button.
3. Enter the name of the new Collaboration Rule, then click **OK** (for this case, use **crLDAPcompareNdelete**).
4. Double-click the new Collaboration Rules Component to edit its properties. The Collaboration Rules Properties window opens.
5. From the **Service** field drop-down box, select **Java**. The **Collaboration Mapping** tab is now enabled, and the **Subscriptions** and **Publications** tabs are disabled.
6. In the **Initialization string** field, enter any required initialization string that the Collaboration Service may require. This field can be left blank.
7. Select the **Collaboration Mapping** tab.
8. Using the **Add Instance** button, create instances to coincide with the Event Types. For this sample, do the following.
9. In the Instance Name column, enter **LDAP** for the instance name.
10. Click **Find,** navigate to and double-click **ldap.xsc**. This adds **ldap.xsc** to the **ETD** column for this instance.
11. In the **Mode** column, select **Out** from the drop–down list box. To access the drop-down list box, click the right portion of the **Mode** field for this instance.
12. In the **Trigger** column, make sure that the checkbox is cleared (no trigger N/A).
13. In the **Manual Publish** column, make sure the checkbox is cleared (no Manual Publish).

Figure 38 Collaboration Rules - crLDAPcompareNdelete Properties Collaboration Mapping



14 Repeat steps 9–13 using the following values:

- Instance Name — **in**
- ETD — **GenericInEvent.xsc**
- Mode — **In**
- Trigger — selected
- Manual Publish - N/A

15 Repeat steps 9–13 again using the following values:

- Instance Name — **out**
- ETD — **GenericOutEvent.xsc**
- Mode — **Out**
- Trigger — N/A
- Manual Publish - clear

The following section describes how the business logic for the Delete sample schema is set up using the Java Collaboration Rules Editor.

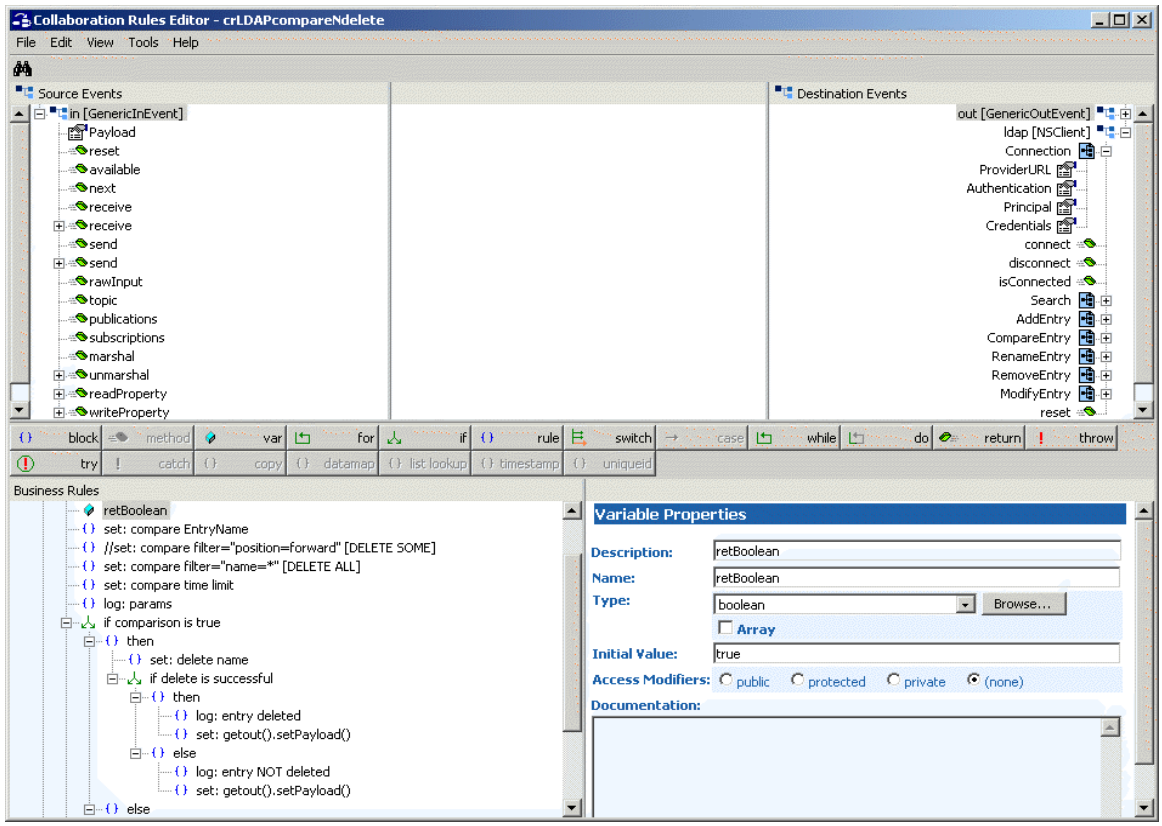
Creating Business Rules for the Delete Sample Schema

The section provides an example of how to create the business rules for the Delete sample schema using the Java Collaboration Rules Editor. The completed Collaboration Rules .xpr file is included with the Delete sample schema on the installation CD-ROM.

Refer to the completed class, **crldapcompareNdelete.class** when completing the Collaboration Rules Properties.

1 Click the **New** or **Edit** button under the Collaboration Rules field. The Java Collaboration Rules Editor opens to the **crLDAPcompareNdelete.xpr** from the Collaboration Rules Properties dialog box. Expand to full size for optimum viewing, expanding the Source and Destination Events as well.

Figure 39 Collaboration Rules Editor crLDAPcompareNdelete



Using the principals of the business logic detailed in the section **"Creating Business Rules Using the Java Collaboration Rules Editor" on page 71**, apply the business logic to create the crLDAPdelete sample schema.

To create the Collaboration for Delete sample schema, follow the steps in **"Creating the Collaborations" on page 74**. For step 8b, use "**colLDAPdelete**" as the source type.

To execute the Delete sample schema, see **"Executing the Schema" on page 78**.

Chapter 6

LDAP e*Way Classes and Methods

This chapter explains the Java classes and methods contained in the e*Way Intelligent Adapter for LDAP, which are used to extend the functionality of the e*Way.

6.1 LDAP e*Way Classes and Methods: Overview

For any e*Way, communication takes place both on the e*Gate system and the external system side. Communication between the e*Way and the e*Gate environment is common to all e*Ways, while the communication between the e*Way and the external system is different for each e*Way.

For the LDAP e*Way, the **stceway.exe** file (Multi-Mode e*Way, see **Chapter 3**) is used to communicate between the e*Way and e*Gate, and a Java Collaboration is utilized to keep the communication open between the e*Way and the LDAP server.

Using Java Methods

Java methods have been added to make it easier to set information in the LDAP e*Way Event Type Definition (ETD), as well as get information from it. The nature of this data transfer depends on the configuration parameters (see **Chapter 3**) you set for the e*Way in the e*Gate Enterprise Manager's e*Way Editor.

The Enterprise Manager's Collaboration Editor allows you to call Java methods by dragging and dropping an ETD node into the **Rules** dialog box.

Note: *The node name can be different from the Java method name.*

After you drag and drop, the actual conversion takes place in the **ldap.xsc** file. To view the **ldap.xsc** file, use the Enterprise Manager's ETD Editor and Collaboration Rules Editor. See **"ETD Structure" on page 29** for more information.

For example, if the node name is **Connection**, the associated **javaName** is **Connection**. If you want to get the node value, use the Java method called **getConnection**. If you want to set the node value, use the Java method called **setConnection**.

Java Classes

The LDAP e*Way contains Java methods that are used to extend the functionality of the e*Way. These methods are contained in the following classes:

6.2 com.stc.eways.jndi.AddAttributesValues Class

This class implements the Add attributes and/or values portion of the ETD to allow for adding attributes and/or values to an existing entry in the directory.

```
java.lang.Object
|
+ - - com.stc.eways.jndi.AddAttributesValues
```

Direct Known Subclasses

```
public class AddAttributesValues
```

extends java.lang.Object

Methods of the AddAttributesValues Class

These methods are described in detail on the following pages:

initialize

Description

Initialize.

Syntax

```
public void initialize (NSClient nsClient)
```

Parameters

Name	Type	Description
nsClient	NSClient	The naming service client object.

Return Values

None.

Throws

None.

getSTCEntry

Description

Gets the entry.

Syntax

```
public STCEntry getSTCEntry()
```

Parameters

None.

Return Values

STCEntry

Throws

None.

getEntryOptions

Description

Gets the options object for setting options.

Syntax

```
public EntryOptions getEntryOptions()
```

Parameters

None.

Return Values

EntryOptions

Throws

None.

performAddAttributesValues

Description

Adds attributes and/or values to an existing entry. If a particular attribute does not exist, the attribute and its values will be added for that entry. If a particular attribute already exists, then it attempts to add the specified values for that attribute.

If a particular attribute already exists and the specified value(s) already exists, then an exception will be thrown. An attribute with no values specified will also result in an exception being thrown.

The entry can be defined by calling getSTCEntry() and setting the appropriate entry name, attributes, and values, prior to calling this method.

Syntax

```
public boolean performAddAttributesValues()
```

Parameters

None.

Return Values

boolean

Throws

com.stc.common.collabService.CollabDataException

reset

Description

Resets and clears everything previously set.

Syntax

```
public void reset()
```

Parameters

None.

Return Values

None.

Throws

None.

6.3 com.stc.eways.jndi.AddEntry Class

This class implements the Add entry portion of the ETD to allow for adding an entry to the directory.

```
java.lang.Object
|
+ - - com.stc.eways.jndi.AddEntry
```

Direct Known Subclasses

```
public class AddEntry
```

extends java.lang.Object

Methods of the AddEntry Class

These methods are described in detail on the following pages:

getAddEntryOptions on page 94
getSTCEntry on page 94
initialize on page 93
performAddEntry on page 94
reset on page 95

initialize

Description

Initialize.

Syntax

```
public void initialize (NSClient nsClient)
```

Parameters

Name	Type	Description
nsClient	NSClient	The naming service client object.

Return Values

None.

Throws

None.

getAddEntryOptions

Description

Gets the add entry options.

Syntax

```
public EntryOptions getEntryOptions()
```

Parameters

None.

Return Values

EntryOptions

Throws

None.

getSTCEntry

Description

Gets the entry.

Syntax

```
public getSTCEntry()
```

Parameters

None.

Return Values

getSTCEntry

Throws

None.

performAddEntry

Description

Adds the entry. The entry can be defined by calling getSTCEntry() and setting the appropriate values for name, attributes, and values, prior to calling this method.

Syntax

```
public boolean performAddEntry()
```

Parameters

None.

Return Values

boolean

Throws

com.stc.common.collabService.CollabDataException

---

reset

Description

Resets and clears everything previously set.

Syntax

```
public void reset()
```

Parameters

None.

Return Values

None.

Throws

None.

6.4 com.stc.eways.jndi.AddEntryOptions Class

This class implements the Add entry options for customizing the characteristics for adding the entry.

```
java.lang.Object
|
+ - - com.stc.eways.jndi.AddEntryOptions
```

Direct Known Subclasses

```
public class AddEntryOptions
```

extends java.lang.Object

Methods of the AddEntryOptions Class

These methods are described in detail on the following pages:

getIgnoreAlreadyBound on page 96 **setIgnoreAlreadyBound** on page 96

getIgnoreAlreadyBound

Description

Gets the boolean flag which indicates whether or not to ignore the NameAlreadyBoundException when attempting to add an entry that already exists.

Syntax

```
public boolean getIgnoreAlreadyBound()
```

Parameters

None.

Return Values

boolean

Throws

None.

setIgnoreAlreadyBound

Description

Sets the boolean flag which indicates whether or not to ignore the NameAlreadyBoundException when attempting to add an entry that already exists.

Syntax

```
public void setIgnoreAlreadyBound(boolean ignoreAlreadyBound)
```

Parameters

None.

Return Values

True.

Throws

None.

6.5 com.stc.eways.jndi.AttributesSelection Class

This class implements the attributes selection list.

```
java.lang.Object
|
+ - - com.stc.eways.jndi.AttributesSelection
```

Direct Known Subclasses

```
public class AttributesSelection
```

extends java.lang.Object

Methods of the AttributesSelection Class

These methods are described in detail on the following pages

addAttribute on page 97
clearAttributes on page 98
removeAttribute on page 97

addAttribute

Description

Adds an attribute to the attributes list for the attributes to be returned after the search.

Syntax

```
public void addAttribute(java.lang.String attribute)
```

Parameters

Name	Type	Description
attribute	java.lang.String	The attribute to add to the list of attribute constraints.

Return Values

None.

Throws

None.

removeAttribute

Description

Removes an attribute from the attributes list for the attributes to be returned after the search.

Syntax

```
public void removeAttribute(java.lang.String attribute)
```

Parameters

Name	Type	Description
attribute	java.lang.String	The attribute to remove from the list of attribute contraints.

Return Values

None.

Throws

None.

clearAttributes

Description

Clears the attributes list so that searches would return all attributes.

Syntax

```
public void clearAttributes()
```

Parameters

None.

Return Values

None.

Throws

None.

6.6 com.stc.eways.jndi.CompareEntry Class

This class implements the Compare entry portion of the ETD to allow for doing entry compares.

```
java.lang.Object
|
+ -- com.stc.eways.jndi.CompareEntry
```

Direct Known Subclasses

```
public class CompareEntry
```

extends java.lang.Object

Methods of the CompareEntry Class

These methods are described in detail on the following pages

getCompareEntryOptions on page 99
initialize on page 98
performCompare on page 99
reset on page 100

initialize

Description

Initialize.

Syntax

```
public void initialize (NSClient nsClient)
```

Parameters

Name	Type	Description
nsClient	NSClient	The naming service client object.

Return Values

None.

Throws

None.

getCompareEntryOptions

Description

Gets the CompareEntryOption object for the compare options.

Syntax

```
public CompareEntryOptions getCompareEntryOptions()
```

Parameters

None.

Return Values

CompareEntryOptions
The compare options as a CompareEntryOptions object.

Throws

None.

performCompare

Description

Determines whether the particular entry specified has an attribute/values pair (defined by filter).

Syntax

```
public boolean performCompare()
```

Parameters

None.

Return Values

boolean

Throws

com.stc.common.collabService.CollabDataException

---

reset

Description

Resets compare.

Syntax

```
public void reset()
```

Parameters

None.

Return Values

None.

Throws

None.

6.7 com.stc.eways.jndi.CompareEntryOptions

This class implements the Compare entry portion of the ETD to allow for testing whether an entry has a specific attribute/values pair.

```
java.lang.Object
|
+ -- com.stc.eways.jndi.CompareEntryOptions
```

Direct Known Subclasses

```
public class CompareEntryOptions
```

extends java.lang.Object

Methods of the CompareEntryOptions Class

These methods are described in detail on the following pages

getCompareFilter on page 102
getEntryName on page 101
getTimeLimit on page 103
setCompareFilter on page 101
setEntryName on page 101
setTimeLimit on page 102

setEntryName

Description

Sets the name of the entry for comparing.

Syntax

```
public void setEntryName(java.lang.String entryName)
```

Parameters

Name	Type	Description
entryName	java.lang.String	The name of the entry for the compare.

Return Values

None.

Throws

None.

getEntryName

Description

Returns the name of the entry for the compare. Returns null if the name was not previously set.

Syntax

```
public java.lang.String getEntryName()
```

Parameters

None.

Return Values

java.lang.String.
Returns the name of the entry for the compare. Returns null if the name was not previously set.

Throws

None.

setCompareFilter

Description

Sets the compare filter for the compare.

Syntax

```
public void setCompareFilter(java.lang.String filter)
```

Parameters

Name	Type	Description
filter	java.lang.String	The filter expression to use for the compare; may not be null.

Return Values

None.

Throws

None.

Example

```
(sn=Doe)
```

getCompareFilter

Description

Gets the compare filter for the compare.

Syntax

```
public java.lang.String getCompareFilter()
```

Parameters

None.

Return Values

java.lang.String.

Throws

None.

setTimeLimit

Description

Sets the timeout in milliseconds for the compare operation.

Syntax

```
public void setTimeLimit(int timeLimitMilliSec)
```

Parameters

Name	Type	Description
timeLimitMilliSec	int	The timeout value in milliseconds. If 0 is specified, then the compare will wait indefinitely; this is the default.

Return Values

integer
Returns the timeout value in milliseconds.

Throws

None.

getTimeLimit

Description

Gets the timeout in milliseconds for the compare operation.

Syntax

```
public int getTimeLimit()
```

Parameters

None.

Return Values

integer
Returns the timeout value in milliseconds.

Throws

None.

6.8 com.stc.eways.jndi.Connection

This class implements the connection properties for connecting to the naming service.

```
java.lang.Object
|
+ -- com.stc.jcsre.cfg.ConnConfigBase
    |
    + -- com.stc.eways.jndi.Connection
```

Direct Known Subclasses

```
public class Connection
```

extends com.stc.jcsre.cfg.ConnConfigBase

Methods of the Connection Class

These methods are described in detail on the following pages

getProviderURL

Description

Gets the Provider URL string to get to the initial context.

Syntax

```
public java.lang.String getProviderURL()
```

Parameters

None.

Return Values

java.lang.String.

Throws

None.

setProviderURL

Description

Sets the Provider URL string to get to the initial context.

Syntax

```
public void setProviderURL(java.lang.String val)
```

Parameters

Name	Type	Description
val	java.lang.String	The Provider URL string.

Return Values

None.

Throws

None.

hasProviderURL

Description

Determines whether the Provider URL was set for the Connection.

Syntax

```
public boolean hasProviderURL()
```

Parameters

None.

Return Values

boolean

Throws

None.

omitProviderURL

Description

Deletes the Provider URL property from the Connection object.

Syntax

```
public void omitProviderURL()
```

Parameters

None.

Return Values

None.

Throws

None.

getAuthentication

Description

Gets the Authentication method for connecting to the naming service.

Syntax

```
public java.lang.String getAuthentication()
```

Parameters

None.

Return Values

java.lang.String.

Throws

None.

setAuthentication

Description

Sets the Authentication method for connecting to the naming service.

Syntax

```
public void setAuthentication(java.lang.String val)
```

Parameters

Name	Type	Description
val	java.lang.String	The Authentication method string.

Return Values

None.

Throws

None.

hasAuthentication

Description

Determines whether the Authentication method was set.

Syntax

```
public boolean hasAuthentication()
```

Parameters

None.

Return Values

boolean

Throws

None.

omitAuthentication

Description

Deletes the Authentication property from the Connection object.

Syntax

```
public void omitAuthentication()
```

Parameters

None.

Return Values

None.

Throws

None.

getPrincipal

Description

Gets the Principal for connecting to the naming service.

Syntax

```
public java.lang.String getPrincipal()
```

Parameters

None.

Return Values

java.lang.String.

Throws

None.

setPrincipal

Description

Sets the Principal for connecting to the naming service.

Syntax

```
public void setPrincipal(java.lang.String val)
```

Parameters

Name	Type	Description
val	java.lang.String	The principal used for authentication when connecting to the naming service.

Return Values

None.

Throws

None.

hasPrincipal

Description

Determines whether the Principal was set.

Syntax

```
public boolean hasPrincipal()
```

Parameters

None.

Return Values

boolean

Throws

None.

omitPrincipal

Description

Deletes the Principal property from the Connection object.

Syntax

```
public void omitPrincipal()
```

Parameters

None.

Return Values

None.

Throws

None.

getCredentials

Description

Gets the Credentials for connecting to the naming service.

Syntax

```
public java.lang.String getCredentials()
```

Parameters

None.

Return Values

java.lang.String.

Throws

com.stc.common.collabService.CollabConnException

setCredentials

Description

Sets the Credentials for connecting to the naming service.

Syntax

```
public void setCredentials(java.lang.String val)
```

Parameters

None.

Return Values

None.

Throws

None.

hasCredentials

Description

Determines whether the Credentials was set.

Syntax

```
public boolean hasCredentials()
```

Parameters

None.

Return Values

boolean

Throws

None.

omitCredentials

Description

Deletes the Credentials property from the Connection object.

Syntax

```
public void omitCredentials()
```

Parameters

None.

Return Values

None.

Throws

None.

6.9 com.stc.eways.jndi.EntryOptions

This class implements the entry options.

```
java.lang.Object
|
+ -- com.stc.eways.jndi.EntryOptions
```

Direct Known Subclasses

```
public class EntryOptions
```

extends java.lang.Object

Methods of the EntryOptions

These methods are described in detail on the following pages

getIgnoreAttributeIDCase

Description

Get the boolean flag which indicates whether or not the case sensitivity for the attributes are ignored.

Syntax

```
public boolean getIgnoreAttributeIDCase()
```

Parameters

None.

Return Values

boolean

Throws

None.

setIgnoreAttributeIDCase

Description

Sets the boolean flag which indicates whether or not the case sensitivity for the attributes are ignored.

Syntax

```
public void setIgnoreAttributeIDCase(boolean ignoreCase)
```

Parameters

Name	Type	Description
ignoreCase	boolean	True if case sensitivity is ignored, false if case sensitivity is not ignored.

Return Values

None.

Throws

None.

getOrderAttributeValues

Description

Get the boolean flag which indicates whether or not the values for each attribute are ordered.

Syntax

```
public boolean getOrderAttributeValues()
```

Parameters

None.

Return Values

boolean

Throws

None.

setOrderAttributeValues

Description

Sets the boolean flag which indicates whether or not the values for each attribute are ordered.

Syntax

```
public void setOrderAttributeValues(boolean orderValues)
```

Parameters

Name	Type	Description
orderValues	boolean	True if case sensitivity is ignored, false if case sensitivity is not ignored.

Return Values

None.

Throws

None.

6.10 com.stc.eways.jndi.LDAPSearchControls

```
java.lang.Object
|
+ -- com.stc.eways.jndi.LDAPSearchControls
```

Direct Known Subclasses

```
public class LDAPSearchControls
```

extends java.lang.Object

Methods of the LDAPSearchControls

These methods are described in detail on the following pages

getPagedResultsControl on page 115
getSortControlAttributes on page 113
removeSortControlAttributes on page 113
removePagedResultsControl on page 114
setSortControlAttributes on page 112
setPagedResultsControl on page 114

setSortControlAttributes

Description

Set the attributes to sort the results on an LDAP search.

Syntax

```
public void setSortControlAttributes(java.lang.String sortAttrs)
```

Parameters

Name	Type	Description
sortAttrs	java.lang.String	A list of attributes to specify the sort order based on the attribute values. Each attribute in the list is separated by a '|' character. For example, "cn|sn" will sort the results based on the 'cn' attribute first and then sort on the 'sn' attribute after sorting on 'cn'.

Return Values

None.

Throws

com.stc.common.collabService.CollabDataException

getSortControlAttributes

Description

Gets the attributes to sort the results on an LDAP search.

Syntax

```
public java.lang.String getSortControlAttributes()
```

Parameters

None.

Return Values

java.lang.String

Throws

None.

removeSortControlAttributes

Description

Removes the sort attributes previously set.

Syntax

```
public void removeSortControlAttributes()
```

Parameters

None.

Return Values

None.

Throws

com.stc.common.collabService.CollabDataException

setPagedResultsControl

Description

Sets the size of the paged results for an LDAP search result.

Syntax

```
public void setPagedResultsControl(int size)
```

Parameters

Name	Type	Description
size	int	An integer for the maximum number of results returned.

None.

Return Values

None.

Throws

com.stc.common.collabService.CollabDataException

removePagedResultsControl

Description

Set the size of the paged results for an LDAP search result.

Syntax

```
public void removePagedResultsControl()
```

Parameters

None.

Return Values

None.

Throws

com.stc.common.collabService.CollabDataException

getPagedResultsControl

Description

Gets the size of the paged results for an LDAP search result.

Syntax

```
public int getPagedResultsControl()
```

Parameters

None.

Return Values

integer
Returns the size of the paged results for an LDAP search result.

Throws

None.

6.11 com.stc.eways.jndi.ModifyEntry

This class implements the Modify Entry portion of the ETD to allow for doing entry modifications to attributes and/or values.

```
java.lang.Object
|
+ -- com.stc.eways.jndi.ModifyEntry
```

Direct Known Subclasses

```
public class LDAPSearchControls
```

extends java.lang.Object

Methods of the Modify Entry

These methods are described in detail on the following pages

initialize

Description

Initialize.

Syntax

```
public void initialize(NSClient nsClient)
```

Parameters

Name	Type	Description
nsClient	NSClient	The naming service client object.

Return Values

None.

Throws

None.

getAddAttributesValues

Description

Get the AddAttributesValues object for specifying the entry and attributes or values for adding.

Syntax

```
public AddAttributesValues getAddAttributesValues()
```

Parameters

None.

Return Values

AddAttributesValues

Throws

None.

getRemoveAttributesValues

Description

Get the RemoveAttributesValues object for specifying the entry and attributes or values for removal.

Syntax

```
public RemoveAttributesValues getRemoveAttributesValues()
```

Parameters

None.

Return Values

RemoveAttributesValues

Throws

None.

getReplaceValues

Description

Get the ReplaceValues object for specifying the entry and attribute values for replacing.

Syntax

```
public ReplaceValues getReplaceValues()
```

Parameters

None.

Return Values

ReplaceValues

Throws

None.

reset

Description

Reset rename.

Syntax

```
public void reset()
```

Parameters

None.

Return Values

None.

Throws

None.

6.12 com.stc.eways.jndi.NSClient

This class implements the ETDExt interface in order to expose the APIs for the JNDI e*Way.

```
java.lang.Object
|
+ -- com.stc.jcsre.SimpleETDImpl
        |
        + -- com.stc.eways.jndi.NSClient
```

Direct Known Subclasses

```
public class NSClient
```

extends com.stc.jcsre.SimpleETDImpl

Methods of the NSClient

These methods are described in detail on the following pages

connect on page 121
disconnect on page 121
getAddEntry on page 122
get$Configuration on page 120
getCompareEntry on page 123
getConnection on page 120
getConnector on page 125
getModifyEntry on page 124
getRemoveEntry on page 124
getRenameEntry on page 123
getSearch on page 122
initialize on page 118
reset on page 124
setConnection on page 120
setConnector on page 125
terminate on page 119

initialize

Description

Called by external (collab service) to initialize the ETD object.

Syntax

```
public void initialize(com.stc.common.collabService.JCollabController
jcollabControl java.lang.String key, int mode)
```

Parameters

Name	Type	Description
cntrCollab	`java.lang.String`	The Java Collaboration Controller object.
key	`java.lang.String`	The key.
mode	`java.lang.String`	The mode.

Return Values

None.

Throws

com.stc.common.collabService.CollabConnException, com.stc.common.collabService.CollabDataException

Specified by

`initialize` in interface `com.stc.jcsre.ETD`

Overrides

`initialize` in class `com.stc.jcsre.SimpleETDImpl`

terminate

Description

Called by external (collab service) prior to terminating the collaboration.

Syntax

```
public void terminate()
```

Parameters

None.

Return Values

None.

Throws

com.stc.common.collabService.CollabConnException

Specified by

`terminate` in interface `com.stc.jcsre.ETD`

Overrides

`terminate` in class `com.stc.jcsre.SimpleETDImpl`

getConnection

Description

Gets the connection instance to allow the user for setting the connection attributes for connecting to the naming service.

Syntax

```
public Connection getConnection()
```

Parameters

None.

Return Values

Connection

Throws

None.

setConnection

Description

Sets the connection instance defining the connection parameters for connecting to the naming service.

Syntax

```
public void setConnection(Connection val)
```

Parameters

Name	Type	Description
Val		Connection object for specifying connection parameters.

Return Values

None.

Throws

None.

get$Configuration

Description

Retrieves the Connector Configuration node object.

Syntax

```
public com.stc.jcsre.cfg.ConnConfigBase get$Configuration()
```

Parameters

None.

Return Values

Connector Configuration node object

Throws

None.

Specified by

`get$Configuration` in interface `com.stc.jcsre.ETDExt`

connect

Description

Manually connects to the LDAP server.

Syntax

```
public void connect()
```

Parameters

None.

Return Values

None.

Throws

com.stc.common.collabService.CollabConnException

disconnect

Description

Manually disconnects from the LDAP server.

Syntax

```
public void disconnect()
```

Parameters

None.

Return Values

None.

Throws

com.stc.common.collabService.CollabConnException

isConnected

Description

Checks the connection to the LDAP server.

Syntax

```
public boolean isConnected()
```

Parameters

None.

Return Values

boolean

Throws

com.stc.common.collabService.CollabConnException

getSearch

Description

Gets the Search object for the search functionality.

Syntax

```
public Search getSearch()
```

Parameters

None.

Return Values

Search
The Search object for searches.

Throws

None.

getAddEntry

Description

Gets the AddEntry object for the add functionality.

Syntax

```
public AddEntry getAddEntry()
```

Parameters

None.

Return Values

AddEntry
The AddEntry object for adding an entry.

Throws

None.

---

getCompareEntry

Description

Gets the CompareEntry object for the compare functionality.

Syntax

```
public CompareEntry getCompareEntry()
```

Parameters

None.

Return Values

CompareEntry
The CompareEntry object for comparing an entry.

Throws

None.

---

getRenameEntry

Description

Gets the RenameEntry object for the rename functionality.

Syntax

```
public RenameEntry getRenameEntry()
```

Parameters

None.

Return Values

RenameEntry
The RenameEntry object for renaming an entry.

Throws

None.

getRemoveEntry

Description

Gets the RemoveEntry object for the remove functionality.

Syntax

```
public RemoveEntry getRemoveEntry()
```

Parameters

None.

Return Values

RemoveEntry
The RemoveEntry object for removing an entry.

Throws

None.

getModifyEntry

Description

Get the ModifyEntry object for the entry modification functionality.

Syntax

```
public ModifyEntry getModifyEntry()
```

Parameters

None.

Return Values

ModifyEntry
The ModifyEntry object for modifying an entry.

Throws

None.

reset

Description

Resets data content of entire ETD.

Syntax

```
public boolean reset()
```

Parameters

None.

Return Values

boolean
True if successfully reset.

Throws

None.

Specified by

`reset` in interface `com.stc.jcsre.ETD`

Overrides

`reset` in class `com.stc.jcsre.SimpleETDImpl`

setConnector

Description

Sets the EBobConnectorExt object that is associated with the ETD.

Syntax

```
public void setConnector(com.stc.jcsre.EBobConnectorExt conn)
```

Parameters

Name	Type	Description
conn	com.stc.jcsre.EbobConnectorExt	The EBobConnectorExt object.

Return Values

None.

Throws

None.

Specified by

`setConnector` in interface `com.stc.jcsre.ETDExt`

getConnector

Description

Retrieves the EBobConnectorExt object that's associated with the ETD.

Syntax

```
public com.stc.jcsre.EBobConnectorExt getConnector()
```

Parameters

None.

Return Values

com.stc.jcsre.EBobConnectorExt
The `EBobConnectorExt` object

Throws

None.

Specified by

`getConnector` in interface `com.stc.jcsre.ETDExt`

6.13 com.stc.eways.jndi.runtime.NSConnector

This class implements the EBobConnector interface in order to provide a connector object for LDAP.

```
java.lang.Object
|
+ -- com.stc.jcsre.EBobConnectorExtImpl
        |
        + -- com.stc.eways.jndi.runtime.NSConnector
```

Direct Known Subclasses

```
public class NSConnector
```

extends com.stc.jcsre.EBobConnectorExtImpl

Methods of the NSConnector

These methods are described in detail on the following pages

open

Description

Opens the connector for accessing the external system.

Syntax

```
public void open(boolean intoEgate)
```

Parameters

None.

Return Values

None.

Overrides

`open` in class `com.stc.jcsre.EBobConnectorExtImpl`

Throws

com.stc.jcsre.EBobConnectionException

open

Description

Opens the connector for accessing the external system.

Syntax

```
public void open(java.util.Properties connectProps)
```

Parameters

Name	Type	Description
connectProps		A Properties object containing the connection parameters for establishing the connection to the external system.

Return Values

None.

Overrides

`open` in class `com.stc.jcsre.EBobConnectorExtImpl`

Throws

com.stc.jcsre.EBobConnectionException

close

Description

Closes the connector to the external system and releases resources.

Syntax

```
public void close()
```

Parameters

None.

Return Values

None.

Overrides

`close` in class `com.stc.jcsre.EBobConnectorExtImpl`

Throws

com.stc.jcsre.EBobConnectionException

---

isOpen

Description

Verifies that the connector to the external system is still available.

Syntax

```
public boolean isOpen()
```

Parameters

None.

Return Values

boolean
True if the connector is still open and available, false if otherwise.

Overrides

`isOpen` in class `com.stc.jcsre.EBobConnectorExtImpl`

Throws

com.stc.jcsre.EBobConnectionException

---

getProperties

Description

Retrieves the connection properties (stored by the constructor) used by the connector to access the external system.

Syntax

```
public java.util.Properties getProperties()
```

Parameters

None.

Return Values

java.util.Properties
Connection properties of the external system.

Overrides

`getProperties` in class `com.stc.jcsre.EBobConnectorExtImpl`

Throws

None.

getContext

Description

Gets the initial context.

Syntax

```
public javax.naming.Context getContext()
```

Parameters

None.

Return Values

javax.naming.Context
A context object for a connection with the naming service.

Overrides

`getProperties` in class `com.stc.jcsre.EBobConnectorExtImpl`

Throws

com.stc.jcsre.EBobConnectionException

setLastError

Description

Sets the most recent error (resulted from an operation).

Syntax

```
public void setLastError(java.lang.Throwable lastError)
```

Parameters

Name	Type	Description
ex		The exception resulted from the last operation.

Return Values

None.

Overrides

`setLastError` in class `com.stc.jcsre.EBobConnectorExtImpl`

Throws

None.

6.14 com.stc.eways.jndi.RCFUtil

This utility class can be used to create and manage a LDAP referral credentials file.

```
java.lang.Object
|
+ -- com.stc.eways.jndi.RCFUtil
```

Direct Known Subclasses

```
public class RCFUtil
```

extends java.lang.Object

6.15 com.stc.eways.jndi.RemoveAttributesValues

This class implements the remove attributes and/or values portion of the ETD to allow for removing attributes and/or values to an existing entry in the directory.

```
java.lang.Object
  |
  +--com.stc.eways.jndi.RemoveAttributesValues
```

Direct Known Subclasses

```
public class RemoveAttributesValues
```

extends java.lang.Object

Methods of the RemoveAttributesValues

These methods are described in detail on the following pages

getEntry on page 131
getEntryOptions on page 131
initialize on page 130
performRemoveAttributesValues on page 132
reset on page 132

initialize

Description

Initialize.

Syntax

```
public void initialize(NSClient nsClient)
```

Parameters

Name	Type	Description
nsClient	NSClient	The naming service client object.

Return Values

None.

Throws

None.

getEntry

Description

Get the entry.

Syntax

```
public STCEntry getEntry()
```

Parameters

None.

Return Values

STCEntry

Throws

None.

getEntryOptions

Description

Get the options object for setting options.

Syntax

```
public EntryOptions getEntryOptions()
```

Parameters

None.

Return Values

EntryOptions

Throws

None.

performRemoveAttributesValues

Description

Removes attributes and/or values from an existing entry. If the specified attribute(s) exists and the specified value(s) for each attribute exists, then the corresponding value(s) will be removed for each of those corresponding attribute(s). Deleting the last value for an attribute, that does not require at least one value, will delete the attribute as well. If the specified attribute(s) exists and at least one specified value for a particular attribute does not exist, then an exception will be thrown and nothing will be removed. If a particular attribute does not have any values specified, then that attribute will be removed if that attribute exists; an exception will be thrown if that attribute does not exist.

Syntax

```
public boolean performRemoveAttributesValues()
```

Parameters

None.

Return Values

boolean

Throws

com.stc.common.collabService.CollabDataException

reset

Description

Reset and clear everything previously set.

Syntax

```
public void reset()
```

Parameters

None.

Return Values

None.

Throws

None.

6.16 com.stc.eways.jndi.RenameEntry

This class implements the Rename entry portion of the ETD to allow for doing entry renames.

```
java.lang.Object
  |
  +--com.stc.eways.jndi.RenameEntry
```

Direct Known Subclasses

```
public class RenameEntry
```

extends java.lang.Object

Methods of the RenameEntry

These methods are described in detail on the following pages

getOldName on page 134
getNewName on page 135
initialize on page 133
performRename on page 135
reset on page 135
setNewName on page 134
setOldName on page 133

initialize

Description

Initialize.

Syntax

```
public void initialize(NSClient nsClient)
```

Parameters

Name	Type	Description
nsClient	NSClient	The naming service client object.

Return Values

None.

Throws

None.

setOldName

Description

Sets the name of the entry to rename.

Syntax

```
public void setOldName(java.lang.String.oldName)
```

Parameters

Name	Type	Description
oldName	java.lang.String	The name of the entry to rename.

Return Values

None.

Throws

None.

getOldName

Description

Gets the name of the entry to rename.

Syntax

```
public java.lang.String getOldName()
```

Parameters

None.

Return Values

java.lang.String

Throws

None.

setNewName

Description

Sets the new name for the entry that is to be renamed.

Syntax

```
public void setNewName(java.lang.String newName)
```

Parameters

Name	Type	Description
newName	java.lang.String	The new name for the entry.

Return Values

None.

Throws

None.

getNewName

Description

Gets the new name for the renamed entry previously set.

Syntax

```
public java.lang.String getNewName()
```

Parameters

None.

Return Values

java.lang.String

Throws

None.

performRename

Description

Renames an entry from the old name to the new name.

Syntax

```
public boolean performRename()
```

Parameters

None.

Return Values

boolean

Throws

com.stc.common.collabService.CollabDataException

reset

Description

Reset rename.

Syntax

```
public void reset()
```

Parameters

None.

Return Values

None.

Throws

None.

6.17 com.stc.eways.jndi.ReplaceValues

This class implements the Replace attribute values portion of the ETD to allow for replacing the values of an attribute of an entry in the directory.

```
java.lang.Object
  |
  +--com.stc.eways.jndi.ReplaceValues
```

Direct Known Subclasses

```
public class ReplaceValues
```

extends java.lang.Object

Methods of the ReplaceValues

These methods are described in detail on the following pages

getEntry on page 137
getEntryOptions on page 137
initialize on page 136
performReplaceValues on page 137
reset on page 138

initialize

Description

Initialize.

Syntax

```
public void initialize(NSClient nsClient)
```

Parameters

Name	Type	Description
nsClient	NSClient	The naming service client object.

Return Values

None.

Throws

None.

getEntry

Description

Gets the entry.

Syntax

```
public STCEntry getEntry()
```

Parameters

None.

Return Values

STCEntry

Throws

None.

getEntryOptions

Description

Get the options object for setting options.

Syntax

```
public EntryOptions getEntryOptions()
```

Parameters

None.

Return Values

EntryOptions

Throws

None.

performReplaceValues

Description

Replaces values for the specified attribute(s). If a particular attribute exists, then all existing values for that attribute will be replaced by the specified value(s). If a particular attribute does not exist, then the side effect is that the attribute will be added along with the specified values. If a particular attribute does not have values defined, then an exception will be thrown. The entry can be defined by calling getSTCEntry() and setting the appropriate entry name, attributes, and values, prior to calling this method.

Syntax

```
public boolean performReplaceValues()
```

Parameters

None.

Return Values

boolean

Throws

com.stc.common.collabService.CollabDataException

---

reset

Description

Reset and clear everything previously set.

Syntax

```
public void reset()
```

Parameters

None.

Return Values

None.

Throws

None.

6.18 com.stc.eways.jndi.Result

This class implements the Result node of the ETD where the next results will be populated.

```
java.lang.Object
  |
  +--com.stc.eways.jndi.Result
```

Direct Known Subclasses

```
public class Result
```

extends java.lang.Object

Methods of the Result

These methods are described in detail on the following pages

setSearchScope on page 145
getAttributesSelection on page 144
getContextName on page 144
setContextName on page 144

getName

Description

Gets the name of the current result.

Syntax

```
public java.lang.String getName()
```

Parameters

None.

Return Values

java.lang.String

Throws

None.

getSTCAttributes

Description

Gets the collection of attributes.

Syntax

```
public STCAttributes getSTCAttributes()
```

Parameters

None.

Return Values

STCAttributes

Throws

None.

getSTCAttribute

Description

Gets the attribute specified by the index position. If one does not exist, then an empty one will be created.

Syntax

```
public STCAttribute getSTCAttribute()
```

Parameters

Name	Type	Description
i	int	The integer index specifying the position of the attribute to return.

Return Values

STCAttribute

Throws

java.lang.IndexOutOfBoundsException

countSTCAttribute

Description

Gets the number of attributes available.

Syntax

```
public int countSTCAttribute()
```

Parameters

None.

Return Values

integer

Throws

None.

6.19 com.stc.eways.jndi.Search

This class implements the Search portion of the ETD to allow for doing directory searches.

```
java.lang.Object
  |
  +--com.stc.eways.jndi.Search
```

Direct Known Subclasses

```
public class Search
```

extends java.lang.Object

Methods of the Search

These methods are described in detail on the following pages

getLDAPSearchControls on page 141 **getSearchOptions** on page 141

initialize

Description

Initialize.

Syntax

```
public void initialize(NSClient nsClient)
```

Parameters

Name	Type	Description
nsClient	NSClient	The naming service client object.

Return Values

None.

Throws

None.

getLDAPSearchControls

Description

If using an LDAP provider, gets the LDAPSearchControls object for the search controls option.

Syntax

```
public LDAPSearchControls getLDAPSearchControls()
```

Parameters

None.

Return Values

LDAPSearchControls
The LDAP search options as a LDAPSearchControls object.

Throws

None.

getSearchOptions

Description

Gets the SearchOption object for the search options.

Syntax

```
public SearchOptions getSearchOptions()
```

Parameters

None.

Return Values

SearchOptions
The search options as a SearchOptions object.

Throws

None.

getSearchResults

Description

Gets the SearchResults object for the search results.

Syntax

```
public SearchResults getSearchResults()
```

Parameters

None.

Return Values

SearchResults
The search results as a SearchResults object.

Throws

None.

performSearch

Description

Performs a directory search for entries in the directory that match the criteria previously defined.

Syntax

```
public void performSearch()
```

Parameters

None.

Return Values

None.

Throws

com.stc.common.collabService.CollabDataException

---

reset

Description

Resets search.

Syntax

```
public void reset()
```

Parameters

None.

Return Values

None.

Throws

None.

6.20 com.stc.eways.jndi.SearchOptions

This class implements the SearchOptions for the Search functionality.

```
java.lang.Object
  |
  +--com.stc.eways.jndi.SearchOptions
```

Direct Known Subclasses

```
public class SearchOptions
```

extends java.lang.Object

Methods of the SearchOptions

These methods are described in detail on the following pages

getAttributesSelection

Description

Gets the attributes selection list previously set or null if not previously set.

Syntax

```
public AttributesSelection getAttributesSelection()
```

Parameters

None.

Return Values

AttributesSelection

Throws

None.

setContextName

Description

Sets the name of the context for the search (where to start searching).

Syntax

```
public void setContextName(java.lang.String contextName)
```

Parameters

Name	Type	Description
name	java.lang.String	The name of the context for the search.

Return Values

None.

Throws

None.

getContextName

Description

Returns the name of the context for the search. Returns null if context name was not previously set.

Syntax

```
public java.lang.String getcontextName()
```

Parameters

None.

Return Values

java.lang.String

Throws

None.

setSearchScope

Description

Sets the search scope for the search.

Syntax

```
public void setSearchScope
```

Parameters

Name	Type	Description
scope	int	The scope of the directory search. Valid values are OBJECT_SCOPE, ONELEVEL_SCOPE, or SUBTREE_SCOPE.

Return Values

None.

Throws

None.

getSearchScope

Description

Gets the search scope for the search.

Syntax

```
public int getSearchScope()
```

Parameters

None.

Return Values

integer

Throws

None.

setSearchFilter

Description

Sets the search filter for the search.

Syntax

```
public void setSearchFilter (java.lang.String filter)
```

Parameters

Name	Type	Description
filter	java.lang.String	The filter expression to use for the search; may not be null. For example, see RFC 2254 for LDAP.

Return Values

None.

Throws

None.

getSearchFilter

Description

Gets the search filter for the search.

Syntax

```
public java.lang.String getSearchFilter()
```

Parameters

None.

Return Values

java.lang.String

Throws

None.

setTimeLimit

Description

Sets the timeout in milliseconds for the search operation.

Syntax

```
public void setTimeLimit (int timeLimitMilliSec)
```

Parameters

Name	Type	Description
timeLimitMilliSec	int	The timeout value in milliseconds. If 0 is specified, then the search will wait indefinitely; this is the default.

Return Values

None.

Throws

None.

getTimeLimit

Description

Gets the timeout in milliseconds for the search operation.

Syntax

```
public int getTimeLimit()
```

Parameters

None.

Return Values

integer

Throws

None.

setCountLimit

Description

Sets the size limitation on the search results for the search operation.

Syntax

```
public void setCountLimit (int timeLimitMilliSec)
```

Parameters

Name	Type	Description
resultsLimit	int	The maximum number of results returned for the search. If 0 is specified, then return all results if there are any results; this is the default.

Return Values

None.

Throws

None.

getCountLimit

Description

Gets the size limitation on the search results for the search operation.

Syntax

```
public int getCountLimit()
```

Parameters

None.

Return Values

integer

Throws

None.

6.21 com.stc.eways.jndi.SearchResults

This class implements the SearchResults for the Search functionality.

```
java.lang.Object
  |
  +--com.stc.eways.jndi.SearchResults
```

Direct Known Subclasses

```
public class SearchResults
```

extends java.lang.Object

Methods of the SearchResults

These methods are described in detail on the following pages

getNextResult on page 150
getResult on page 149
hasResults on page 149
hasMoreResults on page 149

getResult

Description

Gets the current result.

Syntax

```
public Result getResult()
```

Parameters

None.

Return Values

Result.

Throws

None.

hasResults

Description

Checks to see if any results were returned by the search.

Syntax

```
public boolean hasResults()
```

Parameters

None.

Return Values

boolean.

Throws

None.

hasMoreResults

Description

Checks to see if there are still any pending results returned by the search.

Syntax

```
public boolean hasMoreResults()
```

Parameters

None.

Return Values

boolean.

Throws

java.lang.Exception

getNextResult

Description

Gets the next result returned by the search. Populates the results for retrieval.

Syntax

```
public boolean getNextResult()
```

Parameters

None.

Return Values

boolean.

Throws

com.stc.common.collabService.CollabDataException

6.22 com.stc.eways.jndi.STCAttribute

This class implements the Attribute of an Entry.

```
java.lang.Object
  |
  +--com.stc.eways.jndi.STCAttribute
```

Direct Known Subclasses

```
public class STCAttribute
```

extends java.lang.Object

Methods of the STCAttribute

These methods are described in detail on the following pages

countSTCValue on page 152
getName on page 151
getSTCValue on page 152
getSTCValues on page 151

setName on page 151

getSTCValues

Description

Gets the collection of values for the attribute.

Syntax

```
public STCValues getSTCValues()
```

Parameters

None.

Return Values

STCValues

Throws

None.

getName

Description

Gets the name of the attribute.

Syntax

```
public java.lang.String getName()
```

Parameters

None.

Return Values

java.lang.String

Throws

None.

setName

Description

Sets the name of the attribute.

Syntax

```
public void setName (java.lang.String name)
```

Parameters

Name	Type	Description
name	java.lang.String	The name of the attribute.

Return Values

None.

Throws

None.

countSTCValue

Description

Gets the number of values available.

Syntax

```
public int countSTCValue()
```

Parameters

None.

Return Values

integer

Throws

None.

getSTCValue

Description

Gets the value specified by the index position.

Syntax

```
public STCValue getSTCValue (int j)
```

Parameters

Name	Type	Description
j	int	The integer index specifying the position of the value to return.

Return Values

STCValue.

Throws

java.lang.IndexOutOfBoundsException.

6.23 com.stc.eways.jndi.STCAttributes

This class implements the Attribute collection for an entry.

```
java.lang.Object
  |
  +--com.stc.eways.jndi.STCAttributes
```

Direct Known Subclasses

```
public class STCAttributes
```

extends java.lang.Object

Methods of the STCAttributes

These methods are described in detail on the following pages

countSTCAttribute on page 153 **getSTCAttribute** on page 154

countSTCAttribute

Description

Gets the number of attributes available.

Syntax

```
public int countSTCAttribute()
```

Parameters

None.

Return Values

integer

Throws

None.

getSTCAttribute

Description

Gets the attribute specified by the index position. If one does not exist, then an empty one will be created.

Syntax

```
public STCAttribute getSTCAttribute(int i)
```

Parameters

Name	Type	Description
i	int	The integer index specifying the position of the attribute to return.

Return Values

STCAttribute

Throws

java.lang.IndexOutOfBoundsException.

6.24 com.stc.eways.jndi.STCEntry

This class implements an entry.

```
java.lang.Object
  |
  +--com.stc.eways.jndi.STCEntry
```

Direct Known Subclasses

```
public class STCEntry
```

extends java.lang.Object

Methods of the STCEntry

These methods are described in detail on the following pages

countAttribute on page 156
getAttributes on page 155
getName on page 139
setStringValue on page 158
reset on page 138
isByteArray on page 157

setName

Description

Sets the name for the entry.

Syntax

```
public void setName(java.lang.String name)
```

Parameters

Name	Type	Description
name	java.lang.String	The name of the entry.

Return Values

None.

Throws

None.

getName

Description

Gets the name for the entry.

Syntax

```
public java.lang.String getName()
```

Parameters

None.

Return Values

java.lang.String

Throws

None.

getAttributes

Description

Gets the attribute associated with this entry.

Syntax

```
public STCAttributes getSTCAttributes()
```

Parameters

None.

Return Values

STCAttributes

Throws

None.

getSTCAttribute

Description

Gets the attribute specified by the index position. If one does not exist, then an empty one will be created.

Syntax

```
public STCAttribute getSTCAttribute(int i)
```

Parameters

Name	Type	Description
i	int	The integer index specifying the position of the attribute to return.

Return Values

STCAttribute

Throws

java.lang.IndexOutOfBoundsException.

countAttribute

Description

Gets the number of attributes available.

Syntax

```
public int countAttribute()
```

Parameters

None.

Return Values

integer

Throws

None.

reset

Description

Clears entry name and deletes all attributes (and the associated values) from the entry instance.

Syntax

```
public void reset()
```

Parameters

None.

Return Values

None.

Throws

None.

6.25 com.stc.eways.jndi.STCValue

This class implements the value for an attribute.

```
java.lang.Object
  |
  +--com.stc.eways.jndi.STCValue
```

Direct Known Subclasses

```
public class STCValue
```

extends java.lang.Object

Methods of the STCValue

These methods are described in detail on the following pages

isByteArray

Description

Determines whether the value returned is a byte array or a string.

Syntax

```
public boolean isByteArray()
```

Parameters

None.

Return Values

boolean

Throws

java.lang.Exception

---

isString

Description

Determines whether the value returned is a string or a byte array.

Syntax

```
public boolean isString()
```

Parameters

None.

Return Values

boolean

Throws

java.lang.Exception

---

getStringValue

Description

Gets the value that is of type java.lang.String.

Syntax

```
public java.lang.String getStringValue()
```

Parameters

None.

Return Values

java.lang.String

Throws

java.lang.Exception

---

setStringValue

Description

Sets the value that is of type java.lang.String.

Syntax

```
public void setStringValue(java.lang.String strVal)
```

Parameters

Name	Type	Description
strVal	java.lang.String	The value of type String.

Return Values

java.lang.String

Throws

None.

getByteValue

Description

Gets the value that is of type byte array (byte[]).

Syntax

```
public byte[] getByteValue()
```

Parameters

None.

Return Values

byte

Throws

java.lang.Exception

setByteValue

Description

Sets the value that is of type byte array (byte[]).

Syntax

```
public void setByteValue(byte[] byteVal)
```

Parameters

Name	Type	Description
byteVal	byte[]	The value of type byte array.

Return Values

None.

Throws

None.

setValue

Description

Sets the value object.

Syntax

```
public void setValue(java.lang.Object val)
```

Parameters

Name	Type	Description
val	java.lang.String	The value of type Object.

Return Values

None.

Throws

java.lang.Exception

getValue

Description

Gets the value object.

Syntax

```
public java.lang.Object getValue()
```

Parameters

None.

Return Values

java.lang.Object

Throws

java.lang.Exception

6.26 com.stc.eways.jndi.STCValues

This class implements the collection of values for an attribute.

```
java.lang.Object
  |
  +--com.stc.eways.jndi.STCValues
```

Direct Known Subclasses

```
public class STCValues
```

extends java.lang.Object

Methods of the STCValues

These methods are described in detail on the following pages

countSTCValue on page 161 **getSTCValue** on page 161

---

countSTCValue

Description

Gets the number of values available.

Syntax

```
public int countSTCValue()
```

Parameters

None.

Return Values

integer

Throws

None.

---

getSTCValue

Description

Gets the value specified by the index position.

Syntax

```
public STCValue getSTCValue(int j)
```

Parameters

Name	Type	Description
j	int	The integer index specifying the position of the value to return.

Return Values

STCValue

Throws

java.lang.IndexOutOfBoundsException

6.27 com.stc.eways.jndi.StringUtil

This class provides String conveniences utilities.

```
java.lang.Object
  |
  +--com.stc.eways.jndi.StringUtil
```

Direct Known Subclasses

```
public class StringUtil
```

extends java.lang.Object

Methods of the StringUtil

These methods are described in detail on the following pages

toHexString on page 162

toHexString

Description

Converts a byte array to an equivalent Hex String.

Syntax

```
public static java.lang.String toHexString(byte[] b)
```

Parameters

None.

Return Values

java.lang.String

Throws

None.

Index

H

I

J

M

O

P

R

S

T

U

W