Novell Developer Kit

www.novell.com

LDAP CLASSES FOR JAVA*

February 28, 2008

Novell®

Legal Notices

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

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

Contents

Preface

The LDAP Classes for Java enable you to write applications to access, manage, update, and search for information stored in Novell® eDirectory™ and other LDAP-aware directories.

LDAP (Lightweight Directory Access Protocol) is an emerging Internet standard for accessing directory information, that allows LDAP-enabled applications to access multiple directories. LDAP v3 supports such features as secure connections (through SSL and SASL), entry management, schema management, and LDAP controls and extensions for expanding the functionality of LDAP.

These classes are an implementation of IETF draft 18 of The Java LDAP Application Program Interface. This product is supported by Novell Developer Support.

The LDAP Classes for Java are available for the following platforms:

- Windows* (NT*, 95, 98, and 2000, XP)
- NetWare®
- UNIX* (Aix*, Solaris*, Linux*, and HP-UX*)

This guide contains the following sections:

- Concepts
- Tasks
- Controls and Extensions
- LDAP Event Services
- LDAP Default DSML Serialization
- LDAP Tools
- JavaDoc API Reference
- Revision History

Audience

This guide is intended for Java developers who desire to write applications to access, manage, update, and search for information stored in Novell eDirectory and other LDAP-aware directories.

Feedback

We want to hear your comments and suggestions about this manual. Please use the User Comments feature at the bottom of each page of the online documentation and enter your comments there.

Documentation Updates

For the most recent version of the LDAP Classes for Java Guide, visit the LDAP Classes for Java Web site (http://developer.novell.com/ndk/jldap.htm).

Additional Documentation

For the most recent version of NDK guides, see the NDK Download Web site (http://developer.novell.com/ndk/downloadaz.htm).

Documentation Conventions

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

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

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

Concepts

1

This manual assumes that you have a basic understanding of Novell® eDirectory™ and LDAP and eDirectory integration. For more information on these topics, see

- *NDK: Novell eDirectory Technical Overview*
- *NDK: LDAP and eDirectory*

These manuals are available on the NDK Web site (http://developer.novell.com/ndk/doc_jldap.htm), or on the local disk once they have been installed with one of the Novell LDAP packages for Netware® and Windows* (default installation locations are `C:\Novell\NDK\doc\ndslib` and `C:\Novell\NDK\doc\ldapover`).

1.1 Getting Started

The following sections cover a few basic requirements for getting set up and started with the LDAP Classes for Java:

- "Dependencies" on page 9
- "Supported Platforms" on page 10
- "LDAP Classes" on page 10
- "Sample Code" on page 11

1.1.1 Dependencies

In addition to the LDAP Classes for Java, you will need the following:

- ❑ JRE 1.2 or higher, required to run an application.
- ❑ JDK* 1.2 Standard Edition or higher, required to develop an application. (The classes no longer support JDK 1.1.7).
- ❑ JDK 1.5 Standard Edition is also supported.

Optionally, you will need the following to take full advantage of the functionality offered in the classes:

- ❑ Novell eDirectory 8.5 or higher to develop or run applications using the extensions for naming context and replica management.
- ❑ Novell eDirectory 8 or higher if you wish to develop or run applications that use SSL (Secure Socket Layer).
- ❑ A Sun-compliant implementation of JSSE if you wish to develop or run applications that create TLS(SSL) connections. (see Section 1.2, "Integrating SSL with the LDAP Classes," on page 12).
- ❑ Novell eDirectory 8.7 or higher if you wish to develop or run applications that start/stop TLS (Transport Layer Security).

1.1.2 Supported Platforms

The LDAP Classes for Java enable you to write applications to access, manage, update and search for information stored in eDirectory and other LDAP-aware directories.

Server applications must run on the same machine as the LDAP server, whereas a client application can run on any supported machine. The classes currently support development on the following eDirectory server platforms:

- NetWare 6™
- NetWare 5.1™
- NetWare 5™ with SP4 or higher
- Windows NT* server 4.0 with SP 4
- Solaris 2.8*
- Linux* (tested on Red Hat 6.2*, and 7.2*)
- AIX 4.3*
- HP-UX 11.11*

Client applications remotely access directory information stored on an LDAP server. The classes currently support development of such applications on the following client platforms:

- NetWare 6™
- NetWare 5.1™
- NetWare 5™ with SP4 or higher
- Windows NT* workstation 4.0 with SP 3 and SP 4
- Windows 95*
- Windows 98*
- Windows 2000*
- Windows XP*
- Solaris 2.8*
- Linux* (tested on Red Hat 6.2, and 7.2*)
- AIX 4.3*
- HP-UX 11.11*

1.1.3 LDAP Classes

The LDAP Classes for Java includes the following packages. Both of these java packages are included in the `LDAP.jar` file.

Table 1-1 *Package Details*

Package	Description
com.novell.ldap	This package contains the Novell Java Classes for LDAP. It includes the following: ◆ Classes defined by the IETF Java LDAP API Internet Draft ◆ Classes defined by IETF Java LDAP Internet Drafts on controls ◆ Classes supporting SSL authentication ◆ Classes supporting Novell defined extensions ◆ Classes supporting controls for eDirectory operations ◆ Classes providing OID definitions for common syntaxes, attributes, controls, etc. ◆ Classes supporting generation of ASN.1 for customer defined controls or extensions
org.ietf.ldap	This package contains only those classes defined by the current IETF drafts and RFCs, and should be used when binary compatibility with other SDKs are required. It includes the following: ◆ Classes defined by the IETF Java LDAP API Internet Draft ◆ Classes defined by IETF Java LDAP Internet Drafts on controls

You will need to use either the com.novell.ldap package or the org.ietf.ldap package. For information on the required jar files for SSL security, see Section 1.2, “Integrating SSL with the LDAP Classes,” on page 12

1.1.4 Sample Code

The LDAP Classes for Java contain a number of samples demonstrating common operations. These samples are available on the NDK Web site (http://developer.novell.com/ndk/doc/samplecode/jldap_sample/index.htm), or on the local drive after they have been installed (default location is `C:\Novell\NDK\samples\jldap_sample`).

1.1.5 Debug Version

If you build your application with the debug version of the LDAP Classes for Java, you can specify debug options on your Java load line with the following:

```
-Dldap.debug=[option]
```

Replace option with one of the following values:

Table 1-2 *Debug Options*

String Value	Description
TraceAll	Enables all debug tracing.
RawInput	Enables debug tracing of raw input.

String Value	Description
rawOutput	Enables debug tracing of Raw output.
Referrals	Enables debug tracing of referral processing.
Messages	Enables debug tracing of message processing.
APIRequests	Enables debug tracing of API Requests.
BindSemaphore	Enables debug tracing of the bind semaphore.
Controls	Enables debug tracing of Controls.
ASN1	Enables debug tracing of ASN1 encode/decode.
Encoding	Enables debug tracing of BER Encoding.
Decoding	Enables debug tracing of BER Decoding.
Connections	Enables debug tracing of LDAP Connections.
UrlParse	Enables debug tracing of URL parsing.
DumpBuffer	Enables debug display of buffer dumps.
DumpObject	Enables debug display of object dumps.
DumpObjectHierarchy	Enables debug display of object hierarchy dumps.
DumpObjectConstructors	Enables debug display of object constructor dumps.
DumpObjectFields	Enables debug display of object field dumps.
DumpObjectMethods	Enables debug display of object methods dumps.
VMTraceInstructions	Enables VM instruction trace.
VMTraceMethodCalls	Enables display VM method calls.
TraceTLS	Enables display of TLS calls.

For example, specify the following to enable all debug tracing:

```
-Dldap.debug=TraceAll
```

1.2 Integrating SSL with the LDAP Classes

The LDAP Class Libraries for Java perform their own authentication. To authenticate using SSL, the LDAP server must have a certificate to use with SSL, the Java client must have a place to store the certificates, and the LDAP classes must be set up to use SSL. Thus, three components must be set up to use SSL:

- LDAP server
- Java client
- LDAP classes

1.2.1 LDAP Server

The LDAP server must be set up with a digital certificate from a Certificate Authority. See the documentation for *Novell Certificate Server Version 2* (http://developer.novell.com/ndk/doc/ncslib/npki_enu/data/bktitle.html) for information on setting up a certificate on the NetWare server. Once the certificate is stored in eDirectory, configure the LDAP server to use it in the LDAP Server General Page in ConsoleOne. For instructions on this process, see “Configuring LDAP Services for eDirectory” in the Novell Developer Notes (http://support.novell.com/techcenter/articles/dnd19981101.html).

1.2.2 Java Client

The Java client must be JSSE-compliant and have a KeyStore for storing root certificates. The following instructions demonstrate how to configure the Sun JSSE reference implementation as the Java client and demonstrates how to use the KeyTool in JDK 1.2 to create a KeyStore containing the server certificate.

To download the Sun JSSE reference implementation or for additional information on JSSE, see http://java.sun.com/products/jsse/ (http://java.sun.com/products/jsse).

To configure the Sun JSSE security provider perform the following:

TIP: To assist in debugging, pass -D javax.net.debug=all as a command line parameter.

1 From ConsoleOne®, create and export a trusted root certificate (`a .der file`). In this example, the certificate file is named `ssl.der`.

2 Use the KeyTool from JDK 1.2 to create a KeyStore file. If `c:\test\ssl.der` is the certificate filename and `c:\test\sslkey.keystore` is the KeyStore filename, the command would be as follows:

```
keytool -import -file c:\test\ssl.der -keystore
c:\test\sslkey.keystore -alias "type=r.name=sslkey"
```

This command prompts you for a password to use with the KeyStore file.

3 Set the provider in the security object. This can be done statically in the securities properties file (`JDK\jre\lib\security\java.security`) or dynamically using function calls. To set this provider statically, find the following line in the security properties file:

```
security.provider.1=sun.security.provider.Sun
```

After this line, add the following:

```
security.provider.2=com.sun.net.ssl.internal.ssl.Provider
```

Both lines are required in order for the SSL to work correctly.

To set this provider dynamically execute the following code:

```
Security.addProvider (new
com.sun.net.ssl.internal.ssl.Provider());
```

4 Set the trustStore location in the system properties. This can be done as a command line parameter:

```
-Djavax.net.ssl.trustStore=<keystore path + filename>
```

Or dynamically with the following code:

```
System.setProperty("javax.net.ssl.trustStore", <keystore path +
filename>)
```

Using the keystore created in the example above, the `<keystore path + filename>` would be `c:\test\sslkey.keystore`.

1.2.3 LDAP Classes

To integrate the Sun JSSE security provider with the LDAP Classes complete the following steps:

1. Add the following lines to your Java source file:

   ```
   import java.security.Security;
   ```

2. When making the connection with the LDAPConnection class, change the following line:

   ```
   LDAPConnection ld = new LDAPConnection();
   ```

   To:

   ```
   LDAPConnection ld = new LDAPConnection(new
   LDAPSecureSocketFactory());
   ```

   For an example of setting up a java client to use SSL see SSLConnection.java in the LDAP Classes for Java "Sample Code" on page 11.

1.3 Using the LDAP Classes

This section contains general information that is helpful to understand before you begin developing with the LDAP Classes for Java. This section contains information on LDAP connections, asynchronous and synchronous methods, constraints, LDAP messages, and LDAP URLs.

1.3.1 LDAP Connections

The central LDAP class is an LDAPConnection. This class provides methods to establish an authenticated or anonymous connection to an LDAP server, as well as methods to search for, modify, compare, and delete entries in the directory.

The following code demonstrates the use of an LDAPConnection object to connect to an LDAP server:

```
String MY_HOST = "localhost";
int MY_PORT = 389;
LDAPConnection ld = new LDAPConnection();
ld.connect(MY_HOST, MY_PORT);
```

These four lines use the LDAPConnection object to create an anonymous connection to the LDAP server specified by *MY_HOST*, and the port specified by *MY_PORT*. At this point, you may authenticate to the server using the bind method, or perform another operation.

The LDAP connection can also be created when the socket connect timeout value is passed as a parameter to the LDAPConnection constructor. And this parameter value has to be specified in MILLISECONDS.This sets the underlying socket connect timeout value to the passed in parameter. The constructor takes the value as an integer and an exception is thrown if the connection is not alive for the specified time (in milliseconds).

The following code demonstrates the use of an LDAPConnection object to connect to an LDAP server using the LDAPConnection constructor, which sets the socket connect timeout to 20 seconds

(20000 milliseconds) and connect to MY_HOST. The exception will be thrown if the connection to MY_HOST is not alive for 20 seconds.

```
int MY_SOCKET_TIMEOUT = 20000;
LDAPConnection ld = new LDAPConnection(MY_SOCKET_TIMEOUT);
ld.connect(MY_HOST, MY_PORT);
```

The LDAPConnection class also provides methods for managing settings that are specific to the LDAP session (such as limits on the number of results returned or time-out limits). An LDAPConnection object can be cloned, allowing objects to share a single network connection in a thread-safe manner. For a complete list of the methods and values available from LDAPConnection, see the "JavaDoc API Reference" on page 77.

1.3.2 Using Synchronous or Asynchronous Functions

Blocking versus Non-Blocking: The LDAP protocol provides both synchronous and asynchronous functions. For the synchronous search methods you can set the batch size parameter for functionality similar to the asynchronous search methods.

Asynchronous Functions

Asynchronous functions do not block, they return immediately after initiating the operation. One of the Listener class functions is used to retrieve the results.

Asynchronous functions return either an LDAPResponseListener or an LDAPSearchListener object and optionally may take a listener object as input. Several asynchronous operations may be tied to the same listener object.

Synchronous Functions

Synchronous functions with batch size of zero block until all the results have been received from the server. Synchronous search functions with batch size = n: (non-zero) Block until "n" messages have been received from the server, then let enumeration proceed while queueing additional messages.

The default value of the batch size parameter is 1. Thus by default, an enumeration of search results from a synchronous search operation will return messages as they are received from the server. The enumeration will block if no messages are waiting.

Other differences between asynchronous and synchronous operations are detailed in the operation-specific sections, such as exception handling and referral handling.

The "Sample Code" on page 11 contains both asynchronous and synchronous programs.

1.3.3 Using Simple Bind with LDAP v3

Usually when you are just getting started, your application does not require a secure connection. By default, an anonymous connection is made when you connect to the LDAP server.

If you wish to authenticate to perform an operation that requires more than anonymous access, use the LDAPConnection.bind method.

```
ld.bind(loginDN, loginPW);
```

Passing NULL or "" for the loginDN will authenticate anonymously.

The LDAP Classes for Java support SASL mechanisms for authentication. The SASL mechanisms supported are simple, Digest-MD5, NMAS_LOGIN and EXTERNAL.

1.3.4 Clear Text vs. Encrypted Passwords

Before you can make a non-encrypted connection, the LDAP server must be configured to allow clear-text passwords. For information on setting this option see documentation for the LDAP Group Object (http://www.novell.com/documentation/lg/ndsse/ndsseenu/data/fbabcieb.html).

1.3.5 Using Constraints to Control Operations

LDAP constraints are used to control LDAP operations, allowing you to control the way in which operations are performed. Using constraints you can, for example, enable referral handling, set referral hop limits, and set controls to be sent to the server. For a complete listing of available constraints, see the LDAPConstraints object in the "JavaDoc API Reference" on page 77.

Constraints can be set in one of two ways, depending on the desired behavior:

- **Connection-based:** Constraints set on a connection apply to all operations performed on that connection.
- **Operation-based:** Constraints set on an operation apply only for that operation.

Connection-based Constraints

Connection-based constraints are set by creating a constraints object and setting properties for this object using LDAPConstraints methods (for a list see the JavaDoc API Reference). Once these properties are set, the constraints are set for a connection using the LDAPConnection.setConstraints method.

The following is an example using the LDAPConstraints object to set the maximum time a client will wait for an operation to complete. This setting applies for all operations performed on this connection.

```
LDAPConnection lc = new LDAPConnection();
LDAPConstraints cons = lc.getConstraints();
cons.setTimeLimit(5000);
lc.setConstraints(cons);
```

Note that we used the LDAPConnection.getConstraints method to initialize the LDAPConstraints object. This retrieves all constraints previously set for this connection, allowing us to add our new constraint without changing the existing constraints.

Operation-based Constraints

Operation-based constraints are set by creating a constraints object and setting properties, the same way it is done when setting connection-based constraints. However, instead of setting these constraints on the connection, this constraints object is passed to the method called to perform the operation.

The following is an example using the LDAPConstraints object to set the maximum time a client will wait for a single operation to complete.

```
LDAPConnection lc = new LDAPConnection();
LDAPConstraints cons = lc.getConstraints();
```

```
cons.setTimeLimit(5000);
lc.add(entry, cons);
```

1.3.6 LDAP URLs

LDAP URLs provide a uniform method to access information on an LDAP server. Defined in RFC 2255, LDAP URLs begin with the prefix `LDAP://` or `LDAPS://`. The following provides the syntax and descriptions of an LDAP URL.

```
ldap[s]://<hostname>:<port>/
<base_dn>?<attributes>?<scope>?<filter>?<extension>
```

Note that ldaps is a common enhancement used to denote SSL, and is not defined in an RFC.

Table 1-3 *Field descriptions for an LDAP URL*

URL Element	Default Value	Description
hostname	none	DNS name or IP address of the LDAP server.
port	389	Port of the LDAP server.
base_dn	root	Base DN for the LDAP operation.
attributes	all attributes	A comma-delimited list of attributes to return.
scope	base	Search scope.
filter	(objectClass=*)	Search filter.
extension	none	LDAP extended operations.

NOTE: An attribute list is required if you want to provide a scope (even if the attribute list is blank). To return all attributes within a specific scope you must include <base_dn>??<scope>.

The SDK provides an LDAPUrl class to handle LDAP URLs. This class has methods to store, parse, and manage LDAP URLs. For more information see the "JavaDoc API Reference" on page 77.

Using LDAP URLs When Handling Referrals

If you receive an LDAPReferralExeption, you can retrieve a list of referral URLs using the LDAPReferralException.getReferrals method. This method returns an array of LDAP URL Strings, which can be converted to LDAPUrls and passed directly to LDAP searches, or can be examined to determine whether or not you wish to follow the referrals. For more information see "Referral Exceptions" on page 25.

1.3.7 LDAP Messages

The LDAPMessage class represents the base class for LDAP response messages for asynchronous commands.

For all asynchronous operations you are returned a listener object. Methods of this listener object return an LDAPResponse (a subclass of LDAPMessage), which contains the result of the operation (for example, LDAPException.SUCCESS, or LDAPException.INSUFFICIENT_ACCESS_RIGHTS).

When performing an asynchronous search, a number of LDAPMessage objects are returned. These messages can be one of three sub-types:

- LDAPSearchResult represents an entry returned from your search.
- LDAPSearchResultReference contains a search result reference (referral information) to continue your search.
- an LDAPResponse, signals the end of the results.

In your code, you need to determine the message type and handle it appropriately.

For example, you could perform an asynchronous search and receive nine LDAP messages. Seven of these could be LDAPSearchResults, one could be an LDAPSearchResultReference, and the last one is an LDAPResponse. In your code, you set up conditional statements to determine the message type and handle it appropriately. See "Asynchronous Methods" on page 24 for more information.

1.4 Searching the Directory

Searching is performed using the LDAPConnection.search methods. When you perform an LDAP search, you need to specify the following five basic parameters:

Table 1-4 *Basic Parameters for LDAP Search*

Search Base	The dn of the entry where you would like the search to begin. An empty string equals root.
Search Scope	How deep you would like the search to extend down the tree.
Search Filter	Defines which entries are to be returned.
Attribute List	A null-terminated array of strings indicating which attributes to return for each matching entry.
Types Only	A Boolean specifying whether you would like to return only the attribute names or the attribute types and the attribute values.

Search Base

The search base parameter specifies the DN of the entry where you would like to begin the search, such as ou=development,o=acme.

If you would like the search to begin at the tree root pass an empty String.

NOTE: Beginning a search at the tree root is handled differently by the various LDAP server implementations. If your search doesn't return results, read the root DSE to retrieve valid naming contexts for a valid starting point.

Search Scope

The search scope parameter specifies the depth of the search and can be one of four values:

- **SCOPE_BASE.** Only the entry specified as the search base is include in the search. This is used when you already know the DN of the object and you would like to read its attributes. (The read method may also be used to read the values of a single entry).
- **SCOPE_ONE.** Objects one level below the base (but not including the base) are included in the search. If we specified o=acme as our search base, then entries in the o=acme container would be included, but not the object o=acme.
- **SCOPE_SUB.** All objects below the base, including the base itself, are included in the search.
- **SCOPE_SUBORDINATESUBTREE.** All subordinates of the specified base object, but does not include the base object, as the subtree scope does.

Search Filter

The search filter defines the entries that will be returned by the search. If you are looking for all employees with a title of engineer, the search filter would be (title=engineer).

See "Search Filters" on page 21 for detailed information on constructing filters.

Attribute List

This list specifies the attributes you want returned from any found entry. By default, all attributes are returned. If you would like only the e-mail address of the entries that match your search filter, you would specify email address. To return all attributes, specify null.

NOTE: Note that operational attributes are not automatically returned by a search. You must specify any operational attributes you wish to be returned by passing them as part of the attribute list.

If you would like to return all attributes of an entry and some operational attributes, you would pass either * or ALL_USER_ATTRIBUTES with the names of the operational attributes you would like returned. When passed to a search as the attribute list parameter, the following String would return all attributes and the creatorsName operational attribute:

```
String[] attrList = new String[]{"*", "creatorsName"}
```

Types Only

This specifies whether you would like to return the attributes specified by the attribute list, or the attributes and their values (specifying FALSE for the typesOnly parameter returns the attributes and their values).

1.4.1 Synchronous Searching

The synchronous search methods return an LDAPSearchResults object, which is an enumeration of LDAPEntry objects. The next method is used to retrieve the entries and attributes from an LDAPSearchResults object.

The following is an example of a synchronous search:

```
String searchBase = "ou=development,o=acme";
int searchScope = LDAPConnection.SCOPE_BASE;
String searchFilter = "(title=engineer)";
```

```
LDAPSearchResults searchResults =
   lc.search( searchBase,
              searchScope,
              searchFilter,
              null,
               false);
```

This search returns all entries with a title of engineer in the development Organizational Unit, from the acme Organization.

Note that certain pieces of the following example are left out for brevity (such as error handling and some closing brackets). See search.java in the "Sample Code" on page 11 for a complete example.

```
while (searchResults.hasMore()) {
  LDAPEntry nextEntry = null
  nextEntry = searchResults.next();
  System.out.println("\n We found "+nextEntry.getDN());
```

Next we will set up a second while loop to display this entry's attributes:

```
LDAPAttributeSet attributeSet = nextEntry.getAttributeSet();
Enumeration allAttributes = attributeSet.getAttributes();

while(allAttributes.hasMore()) {
  LDAPAttribute attribute =
 (LDAPAttribute)allAttributes.nextElement();
  String attributeName = attribute.getName();
  System.out.println("\t\t"+ attributeName);
```

Finally we will set up the third while loop to display the values of the found attribute. Note that this code assumes that the attribute values can be displayed as strings:

```
Enumeration allValues = attribute.getStringValues();

if(allValues!= null) {
  while(allValues.hasMore()) {
    String Value = (String) allValues.nextElement();
    System.out.println("\t\t\t" + Value);
```

1.4.2 Asynchronous Searching

Asynchronous searches return LDAPSearchListener objects which are checked by the application for LDAPMessages.

The following is an example of an asynchronous search:

```
String searchBase = "ou=development,o=acme";
int searchScope = LDAPConnection.SCOPE_BASE;
String searchFilter = "(title=engineer)";

LDAPSearchListener listener =
   lc.search(searchBase,
               searchScope,
               searchFilter,
               null,
               false
```

```
              (LDAPSearchListener)null,
               (LDAPSearchConstraints)null);
```

Note that instead of returning an LDAPSearchResults object the asynchronous search methods return an LDAPSearchListener. This listener is checked by the application for LDAPMessages:

```
LDAPMessage message;

while ((message = listener.getResponse())!= null) {
```

The LDAPMessage will be one of the following three types:

- **LDAPSearchResult:** An entry returned by your search.
- **LDAPSearchResultReference:** A continuation reference (referral) to continue your search.
- **LDAPResponse:** A response indicating that the search is completed. If this is SUCCESS, the search completed successfully. If the result code is REFERRAL, you receive a list of URL strings. (The LDAPResponse result codes are documented in the LDAPException object in the JavaDoc).

In your code you must determine the message type and handle it appropriately. This is done using instanceof:

```
if (message instanceof LDAPSearchResultReference) {
```

When the message type has been determined you can handle the response appropriately. For a complete example of an asynchronous search see searchas.java in the "Sample Code" on page 11.

1.4.3 Search Filters

The LDAP search filter grammar is specified in RFC 2254 and 2251. The grammar uses ABNF notation.

```
filter = " (" filtercomp ") "
filtercomp = and / or / not / item

and = "&" filterlist
   filterlist = 1*filter

or = "|" filterlist
   filterlist = 1*filter

not = "!" filterlist
   filterlist = 1*filter

item = simple / present / substring / extensible

simple = attr filtertype value
   attr = name | name;binary
   filtertype = equal / approx / greater / less
   value = data valid for the attribute's syntax

equal = "="
approx = "~="
greater = ">="
less = "<="
```

```
present = attr "=*"
   attr = name | name;binary

substing = attr "=" [initial] any [final]
   attr = name | name;binary
   initial = value
   any = "*" *(value "*")
   final = value

extensible = attr [":dn"] [":" matchingrule] ":="value
            / [":dn] ":" matchingrule ":=" value
            / matchingrule = name | OID
```

For additional options for the attr option, see Section 4.1.5 of RFC 2251.

For additional information on the value option, see Section 4.1.6 of RFC 2251.

IMPORTANT: Novell eDirectory does not support LDAP approximate (~=) matching or extensible matching rules.

Operators

Table 1-5 *LDAP Filter Operators*

Operator	Description
=	Used for presence and equality matching. To test if an attribute exists in the directory, use (attributename=*). All entries that have the specified attribute will be returned. To test for equality, use attributename=value. All entries that have attributename=value are returned. For example, (cn=Kim Smith) would return entries with Kim Smith as the common name attribute. (cn=*) would return all entries that contained a cn attribute. The = operator can also be used with wildcards to find a substring, (cn=*ary*) would return mary, hillary, and gary.
>=	Used to return attributes that are greater than or equal to the specified value. For this to work, the matching rule defined by the attribute syntax must have defined a mechanism to make this comparison. For example, (cn>=Kim Smith) would return all entries from Kim Smith to Z.
<=	Used to return attributes that are less than or equal to the specified value. For this to work, the matching rule defined by the attribute syntax must have defined a mechanism to make this comparison. For example, (cn<=Kim Smith) would return all entries from A to Kim Smith.
~=	Used for approximate matching. The algorithm used for approximate matching varies with different LDAP implementations.

The following Boolean operators can be combined with the standard operators to form more complex filters. Note that the Boolean operator syntax used is different in search filters than in the C and Java programming languages, but they are conceptually similar.

Table 1-6 LDAP Filter Boolean Operators

Boolean Operators	Description
&	And. For example, (&amp;(cn=Kim Smith) (telephonenumber=555-5555)) would return entries with common name of Kim Smith and a telephone number of 555-5555.
|	Or. For example, (|(cn=Kim Smith)(cn=Kimberly Smith)) would return entries with common name Kim Smith or Kimberly Smith.
!	Not. For example, (!(cn=Kim Smith) would return entries with any cn other than Kim Smith. Note that the ! operator is unary, i.e. operates only on a single filter expression.

Examples

Filter and Description
(cn = Kim Smith)
Returns entries with a common name of Kim Smith.
(&amp;(cn=Kim Smith)(telephonenumber=555*)(emailaddress=*acme.com))
Returns entries with a common name of Kim Smith, a telephone number that starts with 555, and an e-mail address that ends in acme.com
(!(cn = Chris Jones))
Returns entries that do not have a common name of Chris Jones.
(&(objectClass=inetOrgPerson) (| (sn=Smith) (cn=Chris S*)))
Returns entries that are of type inetOrgPerson with a surname of Smith or a common name beginning with Chris S.
(&(o=acme)(objectclass=Country)(!(|(c=spain)(c=us))
Returns entries that are of type Country from the organization Acme, that are not countries spain or us.

1.4.4 Operational Attributes

Operational attributes are not automatically returned in search results; they must be requested by name in the search operation. For a list of supported operational attributes in Novell eDirectory 8.5, see LDAP Operational Attributes in the *LDAP and eDirectory Integration Guide*. The LDAP servers in releases previous to 8.5 do not support requesting operational attributes in a search operation.

1.5 Exception Handling

There are two types of exceptions in the Java LDAP SDK:

- LDAPException
- LDAPReferralException

Most errors that occur throw an LDAPException. An LDAPException is a general exception including an error message and an LDAP result code. An LDAPException can result from physical problems (such as network errors) as well as problems with LDAP operations (for example, if the LDAP add operation fails because of a duplicate entry).

An LDAPException can also contain a nested exception or a Throwable class with more information about the cause of the error. To retrieve the nested exception or Throwable class, use LDAPException.getCause, which returns the lower-level exception which caused the failure. For example, an IOException with additional information may be returned on a CONNECT_ERROR failure. The various result codes returned from an LDAP request are defined as constants in the LDAPException class.

Depending whether you are using asynchronous or synchronous functions, exceptions are handled differently. These differences are outlined in the following sections.

1.5.1 Synchronous Methods

Synchronous methods throw an LDAPException on result codes other than SUCCESS (0). This eliminates the need to check for errors, therefore a standard try/catch statement can be used to handle exceptions.

The following is an example using the LDAPException.toString method to print error information:

```
try {
      [Attempt an LDAP operation]
}
catch(LDAPException e) {
   System.out.println("Error:" + e.toString());
}
```

To facilitate user feedback during synchronous searches, intermediate search results can be obtained before the entire search operation is completed by specifying the number of entries to return at a time. By calling the LDAPSearchConstraints.setBatchSize method, you can set the number of results to obtain before returning information back to the application. The default setting is 1, allowing results to be obtained as they are received by the server. By setting the batch size to 0, you ensure all the results have been received from the server and stored in local memory before returning to the application.

```
LDAPConnection ld = new LDAPConnection();
LDAPSearchConstraints cons = ld.getSearchConstraints();
cons.setBatchSize(0);
ld.setConstraints(cons);
```

This allows you to enumerate through the results without ever blocking.

1.5.2 Asynchronous Methods

For the asynchronous methods, exceptions are thrown only for local or non-server errors, such as connection errors or parameter errors. For server errors, LDAP result messages are returned as LDAPResponse objects which must be checked by the client for errors using the LDAPResponse.getResultCode method.

The following is an example of error handling with a failed asynchronous search.

NOTE: Before you check the response for an error message you need to determine which type of LDAP message has been returned. See "Asynchronous Searching" on page 20 for additional information.

```
//previously determined that the message is an LDAPResponse
if (message instanceof LDAPResponse)
{
  LDAPResponse response = (LDAPResponse) message;
  int status = response.getResultCode();
  if (status!= LDAPException.SUCCESS)
  {
    System.out.println("Asynchronous search failed.");
    throw new LDAPException(response.getErrorMessage(),
                            status,
                            response.getMatchedDN());
  }
}
```

1.5.3 Referral Exceptions

LDAPReferralExceptions are encountered when performing synchronous LDAP operations. They are derived from LDAPException and contain a list of URL strings corresponding to referrals received on an LDAP operation.

LDAPReferralExceptions are thrown when referral handling is turned off in your application, or if the API attempted to follow a referral and the referral could not be followed. For example, if you have enabled automatic referral handling and the API throws an LDAPReferralException, it means the referral could not be followed and you most likely have incomplete results. The LDAPReferral exception will contain a list of LDAP URL strings the API attempted to follow.

The following is an example of retrieving LDAP URLs from an LDAPReferralException:

```
try {
  [perform an LDAP search operation here]
}
catch(LDAPReferralException e) {
  LDAPUrl urls[] = e.getURLs();
  System.out.println("Referral Exception");
  for(i=0;i < urls.length;i++)
    System.out.println(urls[i];
}
```

For more information on LDAPReferralExceptions read the section on Section 1.6, "Referral Handling in LDAP v3," on page 25.

1.6 Referral Handling in LDAP v3

Because of the distributed nature of directory services, a search sent to an LDAP server on eDirectory has a high probability of returning partial data, and referrals for the rest of the data.

When an LDAP server does not contain the requested data and a referral is necessary, the LDAPGroup object in eDirectory can be configured to handle them in one of four ways:

- Configure eDirectory to return complete data and never referrals to the client (always chain).

- Send referrals to the client only for eDirectory servers that do not support chaining.
- Always send referrals to the client (never chain).
- The client applications can be configured to have the API follow referrals, or the applications can perform their own handling of the referrals.

When you are using the LDAP SDK, note that referrals are handled differently for asynchronous and synchronous requests. Details are outlined in the following sections.

1.6.1 Configuring eDirectory to Return Complete Data

In eDirectory, the LDAP server can be configured to return complete data and not return referrals. This is done through the LDAP Group Object using ConsoleOne. For possible configurations in e-Directory, see the documentation for the LDAP Group Object (http://www.novell.com/documentation/lg/ndsse/ndsseenu/data/fbabcieb.html).

1.6.2 Configuring eDirectory to Return Referrals

The LDAP server in eDirectory can also be configured to return referrals to your application. This is done through the LDAP Group Object using ConsoleOne. For possible configurations in Novell e-Directory, see the documentation for the LDAP Group Object (http://www.novell.com/documentation/lg/ndsse/ndsseenu/data/fbabcieb.html)

1.6.3 Enabling Referral Handling in the Application

To enable referral following, use LDAPConstraints.setReferralFollowing passing TRUE to enable referrals, FALSE (default) to disable referrals. See "Using Constraints to Control Operations" on page 16 for information on setting constraints.

1.6.4 Following Referrals Using Synchronous Requests

When performing synchronous operations, referrals can be followed automatically with or without authentication, or they can be handled manually.

Following Referrals Manually

When referral handling is disabled, an LDAPReferralException is thrown if the search result is a referral or a continuation reference.

You can use LDAPReferralExceptions to follow referrals or continuation references by retrieving the URLs from the LDAPReferralException and manually following them.

If you receive some data and an LDAPReferralException is thrown, this is not an error, most likely the server has returned partial data and a continuation reference for the remaining data. A separate LDAPReferralException is thrown for each continuation reference received during a search.

Following Referrals Automatically as Anonymous

If referral following is enabled, referrals are followed by default using an anonymous bind to the next server. If your application does not require authentication, this default behavior is ideal.

If the server encounters a problem following a referral, an LDAPReferralException is thrown. This exception provides information on the URLs that could not be followed, and it may contain a nested exception or Throwable class with more information on what caused the exception. Be aware that if you receive some data and an LDAPReferralException when using automatic referral handling, you most likely have incomplete results. This does not indicate the end of the data in your enumeration.

Following Referrals Automatically with Authentication

If your application requires more than anonymous authentication, you will need to implement a referral handler. The LDAP SDK provides interfaces your application can implement to provide credentials when following referrals. These interfaces are LDAPAuthHandler and LDAPBindHandler.

- **LDAPAuthHandler:** This interface is the simplest to use. Your application creates an object that implements this interface and the SDK uses this class under-the-covers to authenticate.
- **LDAPBindHandler:** Used to do explicit bind processing on a referral. This interface provides greater control over the bind process when following a referral but requires more work.

LDAPAuthHandler

To use LDAPAuthHandler you must create a class that extends the LDAPAuthHandler interface. The following is an example of an LDAPAuthHandler class:

```
class AuthImpl implements LDAPAuthHandler
{
  private LDAPAuthHandler auth;
  AuthImpl(String dn, byte[] pw)
  {
    auth = new LDAPAuthProvider(dn, pw);
    return;
  }

  public LDAPAuthProvider getAuthProvider(String host, int port)
  {
    return auth;
  }
}
```

For more information on LDAPAuthHandler, see SearchUtil.java in the "Sample Code" on page 11.

LDAPBindHandler

To use LDAPBindHandler you must create a class that extends the LDAPBindHandler interface. The LDAPBindHandler class provides the most flexibility, but you must perform your own bind operation. If the bind is successful, the referral will then be followed automatically.

For more information on LDAP rebind, see SearchUtil.java in the "Sample Code" on page 11.

1.6.5 Following Referrals Using Asynchronous Requests

No mechanism for automatic referral handling is defined for asynchronous searches. To follow referrals you need to handle the referrals manually.

For some background information on how asynchronous operations handle responses from the server, read the sections on “LDAP Messages” on page 17 and “Asynchronous Searching” on page 20.

A good example of performing asynchronous searches is contained in searchas.java in the sample code. This sample sets up a number of conditional statements to determine the message type and handle the message. When a referral is encountered, the URL is simply written out to the screen. If you wanted to follow referrals, you would replace this output with code to retrieve the URLs, and connect and bind to the server specified.

1.6.6 Limiting Referral Hops

Your application can specify the maximum number of referral hops the LDAP classes will follow. For example, suppose you set the limit to 2. Your application performs a search, and the search refers you to the following:

Server A refers you to Server B—Hop 1

Server B refers you to Server C—Hop 2

Server C refers you to Server C—Hop 3

The classes will follow the referral through Server C, but they will not continue to Server D because Server D exceeds the hop limit of 2. They return a result code of LDAP_REFERRAL_LIMIT_EXCEEDED.

To set a referral hop limit use LDAPConstraints.setHopLimit passing the maximum number of hops to follow in sequence during automatic referral handling. The default value is 10. See “Using Constraints to Control Operations” on page 16 for information on setting constraints.

1.6.7 Following Referrals on Non-Search Operations

eDirectory 8.7 can return referrals for non-search operations. These referrals are followed using the same methods outlined in “Enabling Referral Handling in the Application” on page 26.

1.7 Other Sources of LDAP and eDirectory Information

For more information on LDAP and eDirectory, check out the following sources:

- Novell's LDAP Developer's Guide (http://btobsearch.barnesandnoble.com/booksearch/isbnInquiry.asp?isbn=0764547208) This guide contains a number of resources for developing LDAP applications.
- “The IETF LDAP Extensions Group (http://www.ietf.org/html.charters/OLD/ldapext-charter.html).” Contains the latest IETF draft of an LDAP interface for Java, as well as drafts for other LDAP extensions.
- "LDAP Schema for eDirectory." Defines the LDAP v3 schema descriptions and non-standard syntaxes used by Novell Directory Services. (This file is included with the LDAP Classes for Java software download).
- "IETF Draft 17 for a Java API for LDAP". Defines the IETF standard for a Java API for LDAP. (This file is included with the LDAP Classes for Java software download).

- Novell's LDAP site (http://developer.novell.com/edirectory). This site contains links to general LDAP information and to specific information about the Novell LDAP server.
- "Netscape Directory SDK 4.0 for Java Programmer's Guide (http://docs.sun.com/source/816-6402-10/contents.htm)." Parts 1 through 3 give explanations on how to perform basic tasks with LDAP, including authentication, using LDAP controls, searching, and managing entries.

For general eDirectory information, check out the following:

- Novell's site for eDirectory developers (http://developer.novell.com/edirectory).

2 Tasks

This section provides instructions for performing a few of the most common LDAP tasks. A number of complete examples is contained in the LDAP Classes for Java "Sample Code" on page 11.

2.1 Establishing an SSL Connection

To establish an SSL connection, both the client and the LDAP server must be set up to use SSL. For instructions, see Section 1.2, "Integrating SSL with the LDAP Classes," on page 12.

2.2 Adding an Entry

Adding an entry involves four steps:

1. Create the attributes of the entry and add them to an attribute set.
2. Specify the DN of the entry to be created.
3. Create an LDAPEntry object with the DN and the attribute set.
4. Call the LDAPConnection add method to add it to the directory.

The following example demonstrates these procedures. For a complete sample, see AddEntry.java in the "Sample Code" on page 11.

Initially, attributes are created for the entry and the attributes are then added to an LDAPAttributeSet object (step 1):

```
LDAPAttribute attribute = null;
String cn_values[] = {"James Smith", "Jim Smith", "Jimmy Smith"};
attribute = new LDAPAttribute("cn", cn_values);
attributeSet.add(attribute);
```

At this point, any number of attributes may be created and added to the attribute set. This example adds only one attribute for brevity.

When the attribute set is created, the DN of the entry must be specified (step 2), and an LDAPEntry object must be created using the DN we just specified (step 3):

```
String dn = "cn=JSmith," + containerName;
LDAPEntry newEntry = new LDAPEntry(dn, attributeSet);
```

Now our application needs to connect to the LDAP server and call LDAPConnection.add to add the entry (step 4):

```
lc.connect(ldapHost, ldapPort);
lc.bind(ldapVersion, loginDN, password);
lc.add(newEntry);
```

IMPORTANT: Entries have required attributes. These must be included in the LDAPAttributeSet in order for the add to succeed.

2.3 Modifying an Entry

For a complete example of modifying an entry, see ModifyAttrs.java in the "Sample Code" on page 11.

2.4 Modifying a Password

eDirectory has a number of restrictions that prevent password modification. The user can have insufficient rights for the following reasons:

- The user is not a supervisor of the entry.
- The flag that allows user to change the password is false.
- The password unique flag is true and the password supplied matches a previous password.
- A minimum length for the password has been set and the password is too short.
- The user did not supply the old password value with the new value in the same operation.

Passwords in eDirectory are stored as RSA public and private key pairs. The Novell LDAP server uses the userPassword attribute to generate these key pairs for an LDAP client.

- eDirectory 8.17 or higher is required for users to change their own passwords.
- eDirectory 7.xx is required for an administrator to change LDAP user passwords.

If the user has sufficient rights, the process is similar to modifying any attribute of an entry. See AddPassword.java in the "Sample Code" on page 11 for a complete example.

IMPORTANT: The delete/add must be in the same modification set.

2.5 Reading the Root DSE

Reading the Root DSE returns information about support of the following features of the LDAP server:

- LDAP versions (v2 and v3)
- LDAP controls and extensions
- Schema object name

With the schema name, you can then extend the schema or read its definitions. You must establish an LDAP v3 connection to read the DSE.

Reading the Root DSE requires the following steps:

1. Set the search base to an empty string.
2. Set the search filter to objectclass=*
3. Set the search scope to LDAP_SCOPE_BASE.

The following example demonstrates these procedures. For a complete sample, see GetDSE.java in the "Sample Code" on page 11.

Initially the LDAP version needs to be set and connect to the LDAP server.

```
int ldapVersion = LDAPConnection.LDAP_V3;

LDAPConnection lc = new LDAPConnection();
lc.connect(LDAP Host, ldapPort);
lc.bind(ldapVersion, loginDN, password);
```

Next, we need to set the search base, filter, and scope.

```
LDAPSearchResults searchResults = lc.search( "",
                        LDAPConnection.SCOPE_BASE,
                        "(objectclass=*)",
                        returnedAttributes,
                         attributeOnly);
```

This search returns one entry which will be the root DSE.

2.6 Reading the Schema

LDAP presents the schema as an object. The name of this object is obtained from the rootDSE object. For eDirectory servers, it's normally "cn=schema", located at the root. This object has an "objectClasses" attribute containing definitions of all the object classes in the schema, and an "attributeTypes" attribute defining all the attribute types.

The schema may be modified by modifying attribute values within this object. For a complete sample see ListSchema.java in the "Sample Code" on page 11.

NOTE: Novell recommends using the Novell Import Convert Export utility for making schema extensions. This utility allows you to input changes via LDIF files eliminating the need to embed schema changes in an application and allows users to view the schema changes in an easy-to-read format.

Reading the schema requires the following steps:

1. Connect to the LDAP server.
2. Read the schema using the LDAPConnection.fetchSchema method.

   The FetchSchema method automatically uses the name of the schema object from the root DSE.
3. Output the classes and attributes.

Initially, we need to connect to the server:

```
LDAPConnection lc = new LDAPConnection();
lc.connect([LDAP Host], [ldapPort]);
lc.bind(ldapVersion, [loginDN], [password]);
```

Next, we create an LDAPSchema object and fetch the schema:

```
LDAPSchema schema = lc.fetchSchema(getSchema = lc.getSchemaDN());
```

We now use an enumeration to read the attributes and classes from the schema:

```
Enumeration enum = schema.getAttributes()
while(enum.hasMoreElements())
{
  LDAPAttributeSchema attr =
  (LDAPAttributeSchema)enum.nextElement();
  System.out.println(attr.getNames()[0]);
  System.out.println(attr.getSyntaxString());
}
enum = schema.getObjectClasses();

while(enum.hasMoreElements())
{
  LDAPObjectClassSchema objcls
  (LDAPObjectClassSchema)enum.nextElement();
  System.out.println(objcls.getNames()[0]);
}
```

See Chapter 7, “JavaDoc API Reference,” on page 77, for more information on LDAP schema.

3 Controls and Extensions

Controls and Extensions were added to version 3 of the LDAP protocol. In version 2, there was no standard mechanism to extend the protocol, requiring developers to extend the protocol in non-standard ways. In version 3, extensions and controls were defined to provide consistent expansion of the protocol.

NOTE: The *LDAP and eDirectory* (http://developer.novell.com/ndk/doc/ldapover/ldap_enu/data/a3wyu4m.html) guide contains a good introduction to LDAP controls and extensions, and contains information you need to be aware of when using these controls with eDirectory. It is recommended that you read the LDAP Controls and the LDAP Extensions chapters in the integration guide.

3.1 Supported Controls

The following table contains a list of controls supported in the LDAP Classes for Java. For examples using these controls, see the LDAP Classes for Java "Sample Code" on page 11.

OID	Description
1.2.840.113556.1.4.473	Sever-side sort control request
1.2.840.113556.1.4.474	Server-side sort control response
2.16.840.1.113730.3.4.9	Virtual list view request
2.16.840.1.113730.3.4.10	Virtual list view response
2.16.840.1.113730.3.4.3	Persistent search
2.16.840.1.113730.3.4.7	Entry change notification
2.16.840.1.113730.3.4.2	Manage Dsa IT
1.2.840.113556.1.4.319	Paged results control request
1.2.840.113556.1.4.319	Paged results control response

- **Server Side Sort:** Returns results from a search operation in sorted order. This can be used to off-load processing from the client, or if you cannot sort the results for some reason.
- **Vertical List View:** Works in conjunction with the Server Side Sort control to provide a dynamic view of a scrolling list. This works in conjunction with the Server Side Sort control.
- **Persistent Search & Entry Change Notification:** Provides a control to perform a continuous search, notifying the application of changes.
- **Manage Dsa IT:** Causes directory-specific entries, regardless of type, to be treated as normal entries. For an example, see SearchUtil.java in the "Sample Code" on page 11.
- **Paged Results Control:** Server Control to specify how search results are to be returned in pages of specified size, by the server.

3.2 Extensions

The following table contains a list of extensions supported in the LDAP Classes for Java. For examples using these extensions, see the LDAP Classes for Java "Sample Code" on page 11.

Table 3-1 *LDAP Extensions*

OID	Name
2.16.840.1.113719.1.27.100.1	ndsToLdapResponse
2.16.840.1.113719.1.27.100.2	ndsToLdapRequest
2.16.840.1.113719.1.27.100.3	Split Partition Request
2.16.840.1.113719.1.27.100.4	Split Partition Response
2.16.840.1.113719.1.27.100.5	MergePartitionRequest
2.16.840.1.113719.1.27.100.6	MergePartitionResponse
2.16.840.1.113719.1.27.100.7	addReplicaRequest
2.16.840.1.113719.1.27.100.8	addReplicaResponse
2.16.840.1.113719.1.27.100.9	refreshLDAPServerRequest
2.16.840.1.113719.1.27.100.10	refreshLDAPServerResponse
2.16.840.1.113719.1.27.100.11	removeReplicaRequest
2.16.840.1.113719.1.27.100.12	removeReplicaResponse
2.16.840.1.113719.1.27.100.13	PartitionEntryCountRequest
2.16.840.1.113719.1.27.100.14	PartitionEntryCountResponse
2.16.840.1.113719.1.27.100.15	changeReplicaTypeRequest
2.16.840.1.113719.1.27.100.16	changeReplicaTypeResponse
2.16.840.1.113719.1.27.100.17	getReplicaInfoRequest
2.16.840.1.113719.1.27.100.18	getReplicaInfoResponse
2.16.840.1.113719.1.27.100.19	listReplicaRequest
2.16.840.1.113719.1.27.100.20	listReplicaResponse
2.16.840.1.113719.1.27.100.21	receiveAllUpdatesRequest
2.16.840.1.113719.1.27.100.22	receiveAllUpdatesResponse
2.16.840.1.113719.1.27.100.23	sendAllUpdatesRequest
2.16.840.1.113719.1.27.100.24	sendAllUpdatesResponse
2.16.840.1.113719.1.27.100.25	RequestPartitionSyncRequest
2.16.840.1.113719.1.27.100.26	RequestPartitionSyncResponse
2.16.840.1.113719.1.27.100.27	requestSchemaSyncRequest
2.16.840.1.113719.1.27.100.28	requestSchemaSyncResponse

OID	Name
2.16.840.1.113719.1.27.100.29	AbortPartitionOperationRequest
2.16.840.1.113719.1.27.100.30	AbortPartitionOperationResponse
2.16.840.1.113719.1.27.100.31	GetBindDNRequest
2.16.840.1.113719.1.27.100.32	Get Bind DN Response
2.16.840.1.113719.1.27.100.33	getEffectivePrivilegesRequest
2.16.840.1.113719.1.27.100.34	getEffectivePrivilegesResponse
2.16.840.1.113719.1.27.100.35	setReplicationFilterRequest
2.16.840.1.113719.1.27.100.36	setReplicationFilterResponse
2.16.840.1.113719.1.27.100.37	getReplicationFilterRequest
2.16.840.1.113719.1.27.100.38	getReplicationFilterResponse
2.16.840.1.113719.1.27.100.39	CreateOrphanPartitionRequest
2.16.840.1.113719.1.27.100.40	CreateOrphanPartitionResponse
2.16.840.1.113719.1.27.100.41	RemoveOrphanPartitionRequest
2.16.840.1.113719.1.27.100.42	RemoveOrphanPartitionResponse
2.16.840.1.113719.1.142.100.1	LburpStartRequest
2.16.840.1.113719.1.142.100.2	LburpStartResponse
2.16.840.1.113719.1.142.100.6	LburpOperationRequest
2.16.840.1.113719.1.142.100.7	LburpOperationResponse
2.16.840.1.113719.1.142.100.4	LburpEndRequest
2.16.840.1.113719.1.142.100.5	LburpEndResponse
2.16.840.1.113719.1.27.100.96	LDAPBackupRequest
2.16.840.1.113719.1.27.100.97	LDAPBackupResponse
2.16.840.1.113719.1.27.100.98	LDAPRestoreRequest
2.16.840.1.113719.1.27.100.101	LDAPDNStoX500DNRequest
2.16.840.1.113719.1.27.100.102	LDAPDNStoX500DNReply

LDAP Event Services 4

LDAP Event Services provide a way for applications to monitor the activity of eDirectory on an individual server using LDAP. LDAP Event Services are available on eDirectory 8.7 and higher.

4.1 Concepts

LDAP Event Services utilizes the standard LDAP extension mechanism to expose the eDirectory event system. The LDAP Libraries for Java (http://developer.novell.com/ndk/jldap.htm) are enhanced to provide support functions to simplify the use of the event system extension.

The event system extension allows the client to specify the events for which it wants to receive notification. This information is sent in the extension request. If the extension request specifies valid events, the LDAP server keeps the connection open and uses the intermediate extended response to notify the client when events occur. Any data associated with an event is also sent in the response. If an error occurs when processing the extended request or during the subsequent processing of events, the server sends an extended response to the client containing error information and then terminates the processing of the request.

4.1.1 Configuring the eDirectory Event System

The eDirectory Event System extension is configured on a per LDAP server basis using the iManager utility (for information, see the iManager Documentation (http://www.novell.com/documentation/lg/imanager20/index.html)). There are two parameters that need to be set. The "allow event monitoring" parameter will turn event monitoring on or off on that particular server. If event monitoring is turned off, the monitor events request will fail. The second parameter is the maximum event monitoring load for the server. A zero value indicates no load limit. Each event type is assigned an integer valued load factor. The load factor is a representation of the loading effect monitoring this event has on the server relative to all other event types. The load is calculated based on each monitored event's load factor and the number of clients registered for that event.

Client Access Rights to Event Data

Any LDAP client can register to monitor any event. Access restrictions are enforced at the time of notification. If the authenticated client does not have access rights to view all of the information in the event, the event will not be sent. The one exception to this rule is the perpetrator DN. If the client does not have rights to the perpertrator object it will be sent as a zero length string. The event notification will still be sent.

4.2 Event Types

The eDirectory event system supports over 200 events. Each event is identified by an integer eventType and most events have associated event data with additional information about the event. This information is returned in one of several structs depending upon the event type.

The eDirectory event system events are grouped according to the structure of the associated event data. This event grouping is outlined in the following table:

Table 4-1 *Event Types*

Event Type	Description
Entry Events	These events indicate the occurrence of individual entry operations such as creating or deleting an entry. The event data is contained in an EntryEventData Class.
Value Events	These events indicate the occurrence of attribute value operations such as deleting or adding a value. The event data is contained in a ValueEventData Class.
General DS Events	These are general events used to indicate a wide variety of DS operations. A generic structure, GeneralDSEventData Class, is used to hold the event data which needs to be interpreted based on the event type.
Bindery Events	These events occurrence of bindery emulation operations. The event data is contained in a BinderyObjectEventData Class.
Security Equivalence Event	This event indicates an entry's security equivalence vector is being checked. The event data is contained in a SecurityEquivalenceEventData Class.
Network Address Events	The data for these events is contained in a NetworkAddressEventData Class.
Events without Data	This classification includes all events that do not have associated data.

4.2.1 Entry Events

The following table lists the event types that are associated with changes to individual attributes.

Table 4-2 *Entry Events*

Val	Event Type	Description
1	EVT_CREATE_ENTRY	A new eDirectory object has been created. This event does not include className and is set to null.
2	EVT_DELETE_ENTRY	An existing eDirectory object has been deleted.
3	EVT_RENAME_ENTRY	An existing eDirectory object has been renamed.
4	EVT_MOVE_SOURCE_ENTRY	This is the second of two events reported for a move operation. This event specifies the deletion of a eDirectory object from its original location in the Directory tree. (See EVT_MOVE_DEST_ENTRY).
14	EVT_MOVE_DEST_ENTRY	This is the first of two events reported for a move operation. This event specifies the placement of the eDirectory object into its new location in the Directory tree. (See EVT_MOVE_SOURCE_ENTRY.) This generates EVT_ADD_VALUE events for all of the values associated with the object.
15	EVT_DELETE_UNUSED_EXTREF	An unused external reference has been deleted.

Val	Event Type	Description
228	EVT_PRE_DELETE_ENTRY	A pre-delete event posted when an entry is about to be deleted.

4.2.2 Value Events

The following table lists the events that are associated with changes to individual attributes:

Table 4-3 *Value Events*

	Event Type	Structure Returned
5	EVT_ADD_VALUE	A value has been added to an object attribute.
6	EVT_DELETE_VALUE	A value has been deleted from an object attribute.
7	EVT_CLOSE_STREAM	A Stream attribute has been closed.
8	EVT_DELETE_ATTRIBUTE	An attribute has been deleted from an object. This generates EVT_DELETE_VALUE events for values associated with the attribute. The EVT_DELETE_VALUE events occur after the EVT_DELETE_ATTRIBUTE event.

4.2.3 Debug Events

The following table lists the events that are associated with debug events:

Table 4-4 *Debug Events*

	Event Type	Description	Data Returned
26	EVT_DB_AUTHEN	An authentication debug message has been sent.	DebugEventData Class
27	EVT_DB_BACKLINK	A backlink debug message has been sent.	DebugEventData Class
28	EVT_DB_BUFFERS	A request buffer debug message has been sent.	DebugEventData Class
29	EVT_DB_COLL	A collision debug message has been sent.	DebugEventData Class
30	EVT_DB_DSAGENT	A low-level DSAgent debug message has been sent.	DebugEventData Class
31	EVT_DB_EMU	A Bindery emulation debug message has been sent.	DebugEventData Class
32	EVT_DB_FRAGGER	A Fragger debug message has been sent.	DebugEventData Class
33	EVT_DB_INIT	An initialization debug message has been sent.	DebugEventData Class

	Event Type	Description	Data Returned
34	EVT_DB_INSPECTOR	An inspector debug message has been sent.	DebugEventData Class
35	EVT_DB_JANITOR	A Janitor process debug message has been sent.	DebugEventData Class
36	EVT_DB_LIMBER	A Limber process debug message has been sent.	DebugEventData Class
37	EVT_DB_LOCKING	A locking debug message has been sent.	DebugEventData Class
38	EVT_DB_MOVE	A move debug message has been sent.	DebugEventData Class
39	EVT_DB_MIN	A default DSTrace (equivalent to ON) debug message has been sent.	DebugEventData Class
40	EVT_DB_MISC	A miscellaneous debug message has been sent	DebugEventData Class
41	EVT_DB_PART	A partition operations debug message has been sent.	DebugEventData Class
42	EVT_DB_RECMAN	A Record Manager debug message has been sent.	DebugEventData Class
44	EVT_DB_RESNAME	A Resolve Name debug message has been sent.	DebugEventData Class
45	EVT_DB_SAP	A SAP debug message has been sent.	DebugEventData Class
46	EVT_DB_SCHEMA	A schema debug message has been sent.	DebugEventData Class
47	EVT_DB_SKULKER	A synchronization debug message has been sent.	DebugEventData Class
48	EVT_DB_STREAMS	A streams debug message has been sent.	DebugEventData Class
49	EVT_DB_SYNC_IN	An incoming synchronization debug message has been sent.	DebugEventData Class
50	EVT_DB_THREADS	An eDirectory thread scheduling debug message has been sent.	DebugEventData Class
51	EVT_DB_TIMEVECTOR	A time vectors debug message has been sent.	DebugEventData Class
52	EVT_DB_VCLIENT	A virtual client debug message has been sent.	DebugEventData Class
166	EVT_DB_NCPENG	An NCPENG debug message has been sent.	DebugEventData Class
175	EVT_DB_AUDIT	An auditing debug message has been sent.	DebugEventData Class

	Event Type	Description	Data Returned
176	EVT_DB_AUDIT_NCP	An auditing NCP™ debug message has been sent.	DebugEventData Class
177	EVT_DB_AUDIT_SKULK	An auditing debug message concerning synchronization has been sent.	DebugEventData Class
184	EVT_DB-CHANGE_CACHE	A change cache debug message has been sent.	DebugEventData Class
186	EVT_DB_PURGE	A purge debug message has been sent.	DebugEventData Class
189	EVT_DB_CLIENT_BUFFERS	A client buffers debug message has been sent.	DebugEventData Class
190	EVT_DB_WANMAN	A WAN Traffic Manager debug message has been sent	DebugEventData Class
198	EVT_DB_DRL	A Distribute Reference Link (DRL) has been created.	DebugEventData Class
202	EVT_DB_ALLOC	A memory allocation debug message has been generated.	DebugEventData Class
204	EVT_DB_SERVER_PACKET	Not implemented.	DebugEventData Class
207	EVT_DB_OBIT	An obituary debug message has been generated.	DebugEventData Class
209	EVT_DB_SYNC_DETAIL	A synchronization detail debug message has been generated.	DebugEventData Class
210	EVT_DB_CONN_TRACE	A connection trace debug message has been generated.	DebugEventData Class
214	EVT_DB_DIRXML	A DirXML® debug message has been sent.	DebugEventData Class
217	EVT_DB_DIRXML_DRIVERS	A DirXML Drivers debug message has been sent.	DebugEventData Class
218	EVT_DB_NDSMON	A NDSMON debug message has been sent.	DebugEventData Class
220	EVT_DB_DNS	A DNS debug message has been sent.	DebugEventData Class
221	EVT_DB_REPAIR	A DS Repair debug message has been sent.	DebugEventData Class
222	EVT_DB_REPAIR_DEBUG	A Repair Debug debug message has been sent.	DebugEventData Class
225	EVT_DB_SCHEMA_DETAIL	A Schema Detail debug message has been sent.	DebugEventData Class
227	EVT_DB_IN_SYNC_DETAIL	A Sync Detail debug message has been sent.	DebugEventData Class

	Event Type	Description	Data Returned
229	EVT_DB_SSL	An SSL debug message has been sent.	DebugEventData Class
230	EVT_DB_PKI	A PKI debug message has been sent.	DebugEventData Class
231	EVT_DB_HTTPSTK	A HTTPSTK debug message has been sent.	DebugEventData Class
232	EVT_DB_LDAPSTK	An LDAPSTK debug message has been sent.	DebugEventData Class
233	EVT_DB_NICIEXT	A NICI Ext debug message has been sent.	DebugEventData Class
234	EVT_DB_SECRET_STORE	A SecretStore debug message has been sent.	DebugEventData Class
235	EVT_DB_NMAS	A NMAS™ debug message has been sent.	DebugEventData Class
236	EVT_DB_BACKLINK_DETAIL	A Backlink Detail debug message has been sent.	DebugEventData Class
237	EVT_DB_DRL_DETAIL	A DRL debug message has been sent.	DebugEventData Class
238	EVT_DB_OBJECT_PRODUCER	An Object Producer debug message has been sent.	DebugEventData Class
239	EVT_DB_SEARCH	A Search debug message has been sent.	DebugEventData Class
240	EVT_DB_SEARCH_DETAIL	A Search Detail debug message has been sent.	DebugEventData Class
242	EVT_DB_NPKI_API	An NPKI debug message has been sent.	DebugEventData Class

4.2.4 General DS Events

A large number of events are classified as general events. The meaning of the data returned is dependent on the type of the event.

For example, when a EVT_LOGIN (event type 100) occurs, the GeneralDSEventData Class contains several general data fields about the event (dstime, milliseconds, curProcess, and verb). The final two output parameters (intValues[], and strValues[]) contain information specific to the EVT_LOGIN event. The description of this information for each event is contained in the Data Returned column in the following table.

In the following example, each integer and string value from the Data Returned column is paired with its corresponding value in the GeneralDSEventData Class:

```
Integer Values
[0] (corresponds to) intValues[0]
[1] (corresponds to) intValues[1]
...
```

```
String Values:
[0] (corresponds to) strValues[0]
[1] (corresponds to) strValues[1]
...
```

The Data Returned column for the EVT_LOGIN event contains the following:

```
Integer Values
[0] 0 if a non-null password was used, 1 if a null password was used
[1] 0 if a bindery login was used, -1 if an NDS login was performed

String Values:
[0] DN of the parent of the entry that performed the login
[1] DN of the entry that performed the login
```

The data described by Integer Values [0] is contained in the first location in the intValues array, intValues[0]. If you wanted to determine if a non-null password was used during the login, intValues[0] would be checked to determine if it is 1 or 0 (with a value of 0 indicating that a non-null password was used).

Table 4-5 *General DS Events*

	Event Type	Description	Data Returned
53	EVT_AGENT_OPEN_LOCAL	The local Directory agent has been opened.	Integer Values: [0] end state of the open operation String Values: [0] end [1] start [2] audit
54	EVT_AGENT_CLOSE_LOCAL	The local Directory agent has been closed.	Integer Values: [0] the state of the operation String Values: [0] end [1] start
55	EVT_DS_ERR_VIA_BINDERY	An error was returned from the bindery.	Integer Values: [0] error code returned from the bindery
56	EVT_DSA_BAD_VERB	An incorrect verb number was given in a DSAgent request.	Integer Values: [0] bad verb number given to the DSA Request (NCP 104, 2)
57	EVT_DSA_REQUEST_START	A DSAgent request has been started.	Integer Values: [0] verb number (NCP 104, 2)
58	EVT_DSA_REQUEST_END	A DSAgent request has completed.	Integer Values: [0] verb number [1] primary ID [2] request size [3] reply size

	Event Type	Description	Data Returned
59	EVT_MOVE_SUBTREE	A container and its subordinate objects have been moved.	String Values: [0] DN of source container [1] DN of destination container
60	EVT_NO_REPLICA_PTR	A replica exists that has no replica pointer associated with it.	String Values: [0] DN of associated partition object
61	EVT_SYNC_IN_END	Inbound synchronization has finished.	Integer Values: [0] number of entries sent String Values: [0] DN of the server entry associated with the server sending the changes [1] DN of root entry of the partition
62	EVT_BKLINK_SEV	A backlink operation has updated an object's Security Equivalence Vector.	String Values: [0] DN of the updated object
63	EVT_BKLINK_OPERATOR	A backlink operation has changed an object's console operator privileges.	String Values: [0] DN of updated entry [1] DN of server entry for which the privileges were changed
64	EVT_DELETE_SUBTREE	A container and its subordinate objects have been deleted.	Integer Values: [0] number of entries deleted String Values: [0] DN of the root object of the deleted subtree
67	EVT_REFERRAL	A referral has been created.	Integer Values: [0] Referral type String Values: [0] DN of the partition [1] DN of the entry
68	EVT_UPDATE_CLASS_DEF	A schema class definition has been updated.	String Values: [0] Name of updated class
69	EVT_UPDATE_ATTR_DEF	A schema attribute definition has been updated.	String Values: [0] Name of updated or added attribute

	Event Type	Description	Data Returned
70	EVT_LOST_ENTRY	eDirectory has encountered a lost entry. A lost entry is an entry for which updates are being received, but no entry exists on the local server.	Integer Values: [0] Seconds field of entry's timestamp [1] replicaNumber field of entry's timestamp [2] Event field of the entry's timestamp String Values: [0] DN of the entry's parent
71	EVT_PURGE_ENTRY_FAIL	A purge operation on an entry has failed.	String Values: [0] DN of the entry for which the purge operation failed
72	EVT_PURGE_START	A purge operation has started.	Integer Values: [0] Replica type String Values: [0] DN of the partition being purged
73	EVT_PURGE_END	A purge operation has ended.	Integer Values: [0] Number of entries purged [1] Number of values purged String Values: [0] DN of the purged partition
76	EVT_LIMBER_DONE	A Limber operation has completed.	Integer Values: [0] 1 indicates all initialized, 0 indicates not all initialized [1] 1 indicates that a new RDN was found, 0 indicates that a new RDN was not found
77	EVT_SPLIT_DONE	A Split Partition operation has completed.	String Values: [0] DN of the parent partition's root [1] DN of the child partition's root

	Event Type	Description	Data Returned
78	EVT_SYNC_SVR_OUT_START	Outbound synchronization has begun from a particular server.	Integer Values: [0] Replica number [1] Replica state, type, and flags String Values: [0] DN of the associated server entry [1] DN of the partition root
79	EVT_SYNC_SVR_OUT_END	Outbound synchronization from a particular server has finished.	Integer Values: [0] Number of objects sent [1] Number of values sent String Values: [0] DN of the associated server entry [1] DN of the partition root
80	EVT_SYNC_PART_START	Synchronization of a partition has begun.	Integer Values: [0] Partition state [1] Replication type String Values: [0] DN of associated partition entry
81	EVT_SYNC_PART_END	Synchronization of a partition has finished.	Integer Values: [0] 1 indicates all processed, 0 indicates not all processed String Values: [0] DN of the associated partition entry
82	EVT_MOVE_TREE_START	A Move Subtree operation has started.	String Values: [0] DN of the root of the subtree to be moved [1] DN of the destination parent entry [2] DN of the server the operation is starting with

	Event Type	Description	Data Returned
83	EVT_MOVE_TREE_END	A Move Subtree operation has finished.	String Values: [0] DN of the root of the moved subtree [1] DN of the server the operation started from
86	EVT_JOIN_DONE	A Join Partitions operation has completed.	String Values: [0] DN of the parent partition root entry [1] DN of the child partition root entry
87	EVT_PARTITION_LOCKED	A partition has been locked.	String Values: [0] DN of the locked partition
88	EVT_PARTITION_UNLOCKED	A partition has been unlocked.	String Values: [0] DN of the unlocked partition
89	EVT_SCHEMA_SYNC	The schema has been synchronized.	Integer Values: [0] 1 indicates all changes processed, 0 indicates not all changes processed
90	EVT_NAME_COLLISION	A name collision (two entries with the same name) has occurred.	String Values: [0] DN of the original entry [1] DN of the duplicate entry
91	EVT_NLM_LOADED	An NLM™ has been loaded.	Integer Values: [0] Module handle of the loaded NLM.
96	EVT_SERVER_RENAME	A server has been renamed.	String Values: [0] New server name
97	EVT_SYNTHETIC_TIME	To bring eDirectory servers into synchronization, synthetic time has been invoked.	Integer Values: [0] Number of time stamps requested String Values: [0] DN of the root entry of the partition issuing the time stamp [1] DN of the partition
99	EVT_DSA_READ	A Read operation has been performed on an entry.	String Values: [0] DN of read entry

	Event Type	Description	Data Returned
100	EVT_LOGIN	A user has logged in.	Integer Values: [0] 0 if a non-null password was used, 1 if a null password was used [1] 0 if a bindery login was used, -1 if an eDirectory login was performed String Values: [0] DN of the parent [1] DN of the entry
101	EVT_CHGPASS	A user's password has changed.	String Values: [0] DN of the parent entry of the changed entry [1] DN of the entry whose password was changed
102	EVT_LOGOUT	A user has logged out.	String Values: [0] DN of the parent entry of entry that logged out [1] DN of the entry that logged out
103	EVT_ADD_REPLICA	A replica of a partition has been added to a server.	Integer Values: [0] Replica type String Values: [0] DN of the root entry of the partition [1] DN of the server entry [2] Name of the server
104	EVT_REMOVE_REPLICA	A replica of a partition has been removed from a server.	String Values: [0] DN of the root entry of the partition [1] DN of the server entry [2] Name of the server
105	EVT_SPLIT_PARTITION	A partition has been split.	String Values: [0] DN of the root entry of the parent partition [1] DN of the root entry of the new partition [2] Name of the new partition entry

	Event Type	Description	Data Returned
106	EVT_JOIN_PARTITIONS	A parent partition has been joined with a child partition.	String Values: [0] DN of the root entry of the parent partition [1] DN of the root entry of the child partition
107	EVT_CHANGE_REPLICA_TYPE	A partition replica's type has been changed.	Integer Values: [0] Old replica type [1] New replica type String Values: [0] DN of the partition's root entry [1] DN of the target server's entry
108	EVT_REMOVE_ENTRY	An entry has been removed beneath a container.	String Values: [0] DN of the container entry [1] DN of the deleted entry
109	EVT_ABORT_PARTITION_OP	A partition operation has been aborted.	String Values: [0] DN of the partition's parent entry [1] DN of the partition entry
110	EVT_RECV_REPLICA_UPDATES	A replica has received an update during synchronization.	String Values: [0] DN of the replica's root entry
111	EVT_REPAIR_TIMESTAMPS	A replica's time stamps have been repaired.	String Values: [0] DN of the replica's root entry
112	EVT_SEND_REPLICA_UPDATES	A replica has sent an update during synchronization.	String Values: [0] DN of the replica's root entry
113	EVT_VERIFY_PASS	A password has been verified.	String Values: [0] DN of the entry's parent [0] DN of the entry
114	EVT_BACKUP_ENTRY	An entry has been backed up.	String Values: [0] Backed-up entry's DN
115	EVT_RESTORE_ENTRY	An entry has been restored.	String Values: [0] DN of the entry's parent [1] RDN of the entry
116	EVT_DEFINE_ATTR_DEF	An attribute definition has been added to the schema.	String Values: [0] Attribute's name
117	EVT_REMOVE_ATTR_DEF	An attribute definition has been removed from the schema.	String Values: [0] Attribute name

	Event Type	Description	Data Returned
118	EVT_REMOVE_CLASS_DEF	A class definition has been removed from the schema.	String Values: [0] Class name
119	EVT_DEFINE_CLASS_DEF	A class definition has been added to the schema.	String Values: [0] Class name
120	EVT_MODIFY_CLASS_DEF	A class definition has been modified.	String Values: [0] Class name
121	EVT_RESET_DS_COUNTERS	The internal eDirectory counters have been reset.	String Values: [0] DN of the server entry
122	EVT_REMOVE_ENTRY_DIR	A file directory associated with an entry has been removed.	String Values: [0] DN of the entry's parent [1] DN of the entry
123	EVT_COMPARE_ATTR_VALUE	A Compare operation has been performed on an attribute.	String Values: [0] DN of the entry's parent [1] DN of the entry [2] Attribute name
124	EVT_STREAM	A stream attribute has been opened or closed.	Integer Values: [0] 0 if the stream was opened, 1 if the stream was closed [1] requested rights (only present if the stream was opened) String Values: [0] DN of the entry [1] Attribute name
125	EVT_LIST_SUBORDINATES	A List Subordinate Entries operation has been performed on a container object.	String Values: [0] DN of the entry's parent [1] DN of the entry
126	EVT_LIST_CONT_CLASSES	A List Containable Classes operation has been performed on an entry.	String Values: [0] DN of the entry's parent [1] DN of the entry
127	EVT_INSPECT_ENTRY	An Inspect Entry operation has been performed on an entry.	String Values: [0] DN of the entry's parent [1] DN of the entry
128	EVT_RESEND_ENTRY	A Resend Entry operation has been performed on an entry.	String Values: [0] DN of the entry's parent [1] DN of the entry

	Event Type	Description	Data Returned
129	EVT_MUTATE_ENTRY	A Mutate Entry operation has been performed on an entry.	String Values: [0] DN of the entry [1] OID of the new class [2] Name of the new class
130	EVT_MERGE_ENTRIES	Two entries have been merged.	String Values: [0] DN of the parent of the winner entry [1] DN of the winner entry [2] DN of the loser entry
131	EVT_MERGE_TREE	Two eDirectory trees have been merged.	String Values: [0] DN of the root entry
132	EVT_CREATE_SUBREF	A subordinate reference has been created.	String Values: [0] subordinate reference ID
133	EVT_LIST_PARTITIONS	A List Partitions operation has been performed.	String Values: [0] DN of the partitions root entry
134	EVT_READ_ATTR	An entry's attributes have been read.	String Values: [0] DN of the entry [1] Attribute's name
135	EVT_READ_REFERENCES	The references on a given object have been read.	String Values: [0] DN of the entry
136	EVT_UPDATE_REPLICA	An Update Replica operation has been performed on a partition replica.	String Values: [0] DN of the partition's root entry [1] DN of the partition entry
137	EVT_START_UPDATE_REPLICA	A Start Update Replica operation has been performed on a partition replica.	String Values: [0] DN of the partition's root entry
138	EVT_END_UPDATE_REPLICA	An End Update Replica operation has been performed on a partition replica.	String Values: [0] DN of the partition's root entry
139	EVT_SYNC_PARTITION	A Synchronize Partition operation has been performed on a partition replica.	String Values: [0] DN of the partition's root entry

	Event Type	Description	Data Returned
141	EVT_CREATE_BACKLINK	A backlink has been created.	Integer Values: [0] Remote entry ID String Values: [0] DN of the server entry making the request [1] DN of the local entry
142	EVT_CHECK_CONSOLE_OPERATOR	An object has been checked for Console Operator rights.	Integer Values: [0] 0 if the entry does not have console rights, 1 if it does String Values: [0] DN of server entry [1] DN of entry being checked
143	EVT_CHANGE_TREE_NAME	The tree name has been changed.	String Values: [0] New tree name
144	EVT_START_JOIN	A Start Join operation has been performed.	String Values: [0] DN of the parent partition's root entry [1] DN of the child partition's root entry
145	EVT_ABORT_JOIN	A Join operation has been aborted.	String Values: [0] DN of the parent partition root [1] DN of the child partition root
146	EVT_UPDATE_SCHEMA	An Update Schema operation has been performed.	String Values: [0] DN of the server entry
147	EVT_START_UPDATE_SCHEMA	A Start Update Schema operation has been performed.	String Values: [0] Name of the Tree root [1] DN of the server entry
148	EVT_END_UPDATE_SCHEMA	An End Update Schema operation has been performed.	String Values: New: [0] Name of the Tree root [1] DN of the server entry

	Event Type	Description	Data Returned
149	EVT_MOVE_TREE	A Move Tree operation has been performed.	Integer Values: [0] Type of string. String Values: [0] DN of the source parent [1] DN of the destination parent [2] If integer[0] = 0, Source DN. If integer [0] = 1, new name.
150	EVT_RELOAD_DS	DS has been reloaded.	String Values: [0] DN of tree root.
151	EVT_ADD_PROPERTY	An attribute (property) has been added to an object.	Integer Values: [0] Security [1] Flags String Values: [0] DN of the entry
152	EVT_DELETE_PROPERTY	An attribute (property) has been removed from an object.	String Values: [0] DN of the entry
153	EVT_ADD_MEMBER	A member has been added to a Group object.	String Values: [0] DN of the group entry [1] DN of the new member [2] Attribute name
154	EVT_DELETE_MEMBER	A member has been deleted from a Group object.	String Values: [0] DN of the group entry [1] DN of the deleted entry [2] Attribute name
155	EVT_CHANGE_PROP_SECURITY	Security for a bindery object's property has been changed.	Integer Values: [0] New security String Values: [0] DN of the object [1] Property's name
156	EVT_CHANGE_OBJ SECURITY	A bindery object's security has been changed.	Integer Values: [0] New security String Values: [0] DN of the object's parent [1] DN of the object

	Event Type	Description	Data Returned
159	EVT_SEARCH	A Search operation has been performed.	Integer Values: [0] Scope [1] Indicates the nodes to search [2] Info type String Values: [0] DN of the base entry
160	EVT_PARTITION_STATE_CHG	A partition's state has changed.	Integer Values: [0] Function [1] Type [2] State String Values: [0] DN of the partition's root entry [1] DN of the partner partition entry
161	EVT_REMOVE_BACKLINK	A backlink has been removed.	String Values: [0] DN of the affected object [1] DN of the affected server entry [2] DN of the remote server entry
162	EVT_LOW_LEVEL_JOIN	A low-level join has been performed.	String Values: [0] DN of the parent's root entry [1] DN of the child partition's root entry
164	EVT_CHANGE_SECURITY_EQUALS	An object's Security Equals attribute has been changed.	Integer Values: [0] 0=delete equivalence, 1=add equivalence String Values: [0] DN of the object whose security has changed. [1] DN of the equivalent object.
167	EVT_CRC_FAILURE	A CRC failure has occurred when fragmented NCP requests were reconstructed.	Integer Values: [0] Failure type, 0=server, 1=client [1] Server/Client CRC error count
168	EVT_ADD_ENTRY	A new object has been added under a container object.	String Values: [0] Container entry's DN [1] Entry's DN

	Event Type	Description	Data Returned
169	EVT_MODIFY_ENTRY	An attribute has been modified on an object.	String Values: [0] Dn of the entry's parent entry [1] Entry's DN
178	EVT_MODIFY_RDN	A Modify RDN operation has been performed.	String Values: [0] Dn of the entry's parent entry [1] Entry's DN [2] Entry's former DN
181	EVT_ENTRYID_SWAP	A Swap Entry ID operation has been performed.	String Values: [0] DN of the source entry [1] DN of the destination entry
185	EVT_LOW_LEVEL_SPLIT	A low-level partition split has been performed.	String Values: [0] DN of the parent partition root entry [1] DN of the child partition root entry
188	EVT_ALLOW_LOGIN	A user has been allowed to log in.	Integer Values: [0] Flags String Values: [0] Entry's DN

4.2.5 Events Without Data

The following events do not have any associated data. When these events occur, the eventData field of the EventMonitorResponse is not present.

Table 4-6 *Events Without Data*

	Event	Description
9	EVT_SET_BINDERY_CONTEXT	A bindery context has been set.
13	EVT_UPDATE_SEV	security equivalence vector is updated.
94	EVT_LUMBER_DONE	Signals that a lumber operation has finished.
95	EVT_BACKLINK_PROC_DONE	Signals that a backlink process has finished.
98	EVT_SERVER_ADDRESS_CHANGE	Signals that a server address has changed.
140	EVT_SYNC_SCHEMA	Signals that the schema has been synchronized.
150	EVT_RELOAD_DS	Signals that eDirectory has been reloaded.
163	EVT_CREATE_NAMEBASE	Signals that a directory namebase has been created.
171	EVT_OPEN_BINDERY	Signals that the bindery has been opened.

	Event	Description
172	EVT_CLOSE_BINDERY	Signals that the bindery has been closed.
174	EVT_NEW_SCHEMA_EPOCH	Signals that a new schema epoch has been declared.
182	EVT_INSIDE_NCP_REQUEST	Entered into NCP request.
187	EVT_END_NAMEBASE_TRANSACTION	A namebase transaction has ended.
213	EVT_BEGIN_NAMEBASE_TRANSACTION	A namebase transaction has begun.

4.2.6 Bindery Events

The following table lists the events that are associated with bindery events:

Table 4-7 *Bindery Events*

	Event	Description
10	EVT_CREATE_BINDERY_OBJECT	Signals that a bindery object has been created.
11	EVT_DELETE_BINDERY_OBJECT	Signals that a bindery object has been deleted.

4.2.7 Security Equivalence Event

The security equivalence event is indicated by the following eventType value:

Table 4-8 *Security Equivalence Event*

	Event	Description
12	EVT_CHECK_SEV	A security equivalence vector has been verified.

4.2.8 Module State Events

The following table lists the events that are associated with module state events:

Table 4-9 *Module State Events*

	Event	Description
21	EVT_CHANGE_MODULE_STATE	A Module state change has happened.

4.2.9 Network Address Events

The following table lists the events that are associated with network address events:

Table 4-10 *Network Address Events*

	Event	Description
17	EVT_REMOTE_SERVER_DOWN	A remote server is down.
18	EVT_NCP_RETRY_EXPENDED	An NCP retry is applied.
158	EVT_CONNECT_TO_ADDRESS	A connection has been established with a particular address. Data returned: Integer Values: [0] taskID [1] Address type [2] Address_size String Values: [0] Address

4.2.10 Connection Change Events

The following table lists the events that are associated with connection change events:

Table 4-11 *Connection Change Events*

	Event	Description
173	EVT_CHANGE_CONN_STATE	Signals that the connection state has changed
212	EVT_COMPUTE_CONN_SEV_INLINE	Connection inline with security equivalence vector has been computed.

4.2.11 Change Server Address

The following table lists the events that are associated with change server address events:

Table 4-12 *Change Server Address*

	Event	Description
219	EVT_CHANGE_SERVER_ADDRS	Server address has been changed.

4.3 Classes

This section explains the Event Data classes used by LDAP Event Services.

4.3.1 Entry Events

This section explains the entry events.

EntryEventData Class

The response data for entry events is returned as an EntryEventData class. This class consists of the read-only properties as explained in the table below.

Table 4-13 *EntryEventData Class*

Property	Type	Description
perpetratorDN	LDAPDN	Specifies the DN of the entry that caused the event.
entry	LDAPDN	Specifies the DN of the entry that was acted upon.
class	OCTET STRING	Specifies the class OID of the object that was acted upon.
creationTime	DSETimestamp	Specifies the Time of Creation of this entry.
verb	INTEGER	Specifies the action that caused the event to occur.
flags	INTEGER	
newDN	OCTET STRING	Specifies the new DN of the entry that was acted upon.

DSETimestamp Class

The class represents the time stamp data structure for eDirectory Events Notification. It contains the time (in seconds), replica number and event type.

Table 4-14 *DSETimestamp Class*

Property	Type	Description
seconds	INTEGER	Specifies in seconds when the event occurred. Zero equals 12:00 midnight, January 1, 1970,
replicaNumber	INTEGER	Specifies the number of the replica on which the change or event occurred.
event	INTEGER	Specifies an integer that further orders events occurring within the same whole-second interval.

Remarks

Two time stamp values are compared by comparing the seconds fields first and the event fields second. If the seconds fields are unequal, order is determined by the seconds field alone. If the seconds fields are equal, and the event fields are unequal, order is determined by the event fields. If the seconds and the event fields are equal, the time stamps are equal.

4.3.2 Value Events

This section details the value events.

ValueEventData Class

The response data for value events is returned as an ValueEventData class. This class consists of the read-only properties as explained in the table below.

Table 4-15 *ValueEventData Class*

Property	Type	Description
perpetratorDN	LDAPDN	Specifies the DN of the entry that caused the event.
entry	LDAPDN	Specifies the DN of the entry that was acted upon.
attribute	OCTET STRING	Specifies the attribute OID of the attribute that was acted upon.
syntax	OCTET STRING	Specifies the Syntax OID of the entry that was acted upon.
class	OCTET STRING	Specifies the class OID of the object that was acted upon.
timeStamp	DSETimestamp	Specifies a TimeStamp.
data	OCTET STRING	Specifies the information that further identifies the changes that were made.
verb	INTEGER	Specifies the action that caused the event to occur.

4.3.3 Debug Events

This section details about the debug events.

DebugEventData Class

The response data for debug event is returned as a DebugEventData class. This class consists of the read-only properties as explained in the table below.

Table 4-16 *DebugEventData Class*

Property	Type	Description
dsTime	INTEGER	Specifies the time the event occurred as the number of seconds elapsed since midnight (00:00:00), January 1, 1970, coordinated universal time, according to the system clock.
milliseconds	INTEGER	The millisecond portion of the time the event occurred.
perpetratorDN	LDAPDN	The DN of the object that caused this event.
formatString	String	The format string used to create the string printed in the DS Trace utility. The format string describes the string that is displayed by the DS Trace utility. It contains literal characters as well as format characters that serve as place holder for parameter values. See the remarks for a list of valid format characters.
verb	INTEGER	The ID of the ds verb that was executing when the event occurred.
Parametercount	INTEGER	Number of elements in the parameters array.

Property	Type	Description
Parameters	Array of DebugParameter	A list of DebugParameter class. The parameters are in the same order as the parameter characters in the format string.

DebugParameter Class

This class contains the debug parameters associated with debug events.

Table 4-17 *DebugParameter Class*

Property	Type	Design
Type	INTEGER	An integer that indicates the type of the parameter. See the Table Below.
Data	Object	Specifies different type of Object depending on Type.

Table 4-18 *Value of Type*

Value of Type	Data Class contained as Data property	Description
DB_PARAM_TYPE_ENTRYID	INTEGER	Contains an Integer Object.
DB_PARAM_TYPE_STRING	String	Contains an UTF-8 encoded string value of the parameter
DB_PARAM_TYPE_BINARY	byte[]	Contains a byte array as data.
DB_PARAM_TYPE_INTEGER	INTEGER	Contains an Integer Object.
DB_PARAM_TYPE_ADDRESS	ReferralAddress	Contains the Network Address contained in a ReferralAddress class.
DB_PARAM_TYPE_TIMESTAMP	DSETimeStamp	Specifies a TimeStamp as a DSETimeStamp class.
DB_PARAM_TYPE_TIMEVECTOR	List of DSETimeStamp	Specifies a list of DSETimeStamps.

Remarks

The formatString parameter is formatted according to the following:

```
%[flags][width][.precision][L,l,h,!]type
```

Table 4-19 *Details of the formatString Parameter*

Element	Description
flags	-, +, #, 0
width	An optional integer indicating the width of the displayed value

Element	Description
precision	An optional integer indicating the precision of the displayed value
L, l, h, !	A character indication the size of the parameter, one of the following values: ◆ L: DOUBLE_FLAG ◆ l: LONG_FLAG ◆ h: SHORT_FLAG ◆ !: I64_FLAG
type	A character indicating the data type of the parameter, one of the following values: ◆ C: color (no associated parameter) ◆ t: current time (no associated parameter) ◆ s: string, EVT_TAG_DB_STRING ◆ a: network address ◆ U: string, EVT_TAG_DB_STRING ◆ T: time stamp ◆ V: time stamp vector ◆ S: string, EVT_TAG_DB_STRING ◆ D: binary data ◆ x: hex integer, EVT_TAG_DB_INTEGER ◆ v: verb number, EVT_TAG_DB_INTEGER ◆ o: octal integer, EVT_TAG_DB_INTEGER ◆ e: error code value, EVT_TAG_DB_INTEGER ◆ d: normal decimal integer, EVT_TAG_DB_INTEGER ◆ c: single character, EVT_TAG_DB_INTEGER ◆ p: raw memory pointer, EVT_TAG_DB_INTEGER ◆ X: HEX integer, EVT_TAG_DB_INTEGER ◆ E: error code value, EVT_TAG_DB_INTEGER

ReferralAddress Class

This class is used to store the network address.

Table 4-20 *ReferralAddress Class*

Property	Type	Description
Type	INTEGER	An integer value indicating the address type.
Address	String	The actual address value.

4.3.4 General DS Events

This section details the general DS events.

GeneralDSEventData Class

The response data for general DS events is returned as a GeneralDSEventData class. This class consists of the read-only properties as explained in the table below.

Table 4-21 *GeneralDSEventData Class*

Property	Type	Description
dsTime	INTEGER	Specifies the time in milliseconds when the event occurred.
milliseconds	INTEGER	
verb	INTEGER	Specifies the action that caused the event to occur.
currentProcess	INTEGER	Specifies the process that was running when the event occurred.
perpetratorDN	LDAPDN	Specifies the DN of the entry that caused the event.
integerValues	Array of integer	Contains event data determined by the event type
Stringvalues	Array of string	Contains event data determined by the event type.

Events Without Data

In the case of events without data, the response data is returned as a null object.

4.3.5 Bindery Events

This section details the bindery events.

BinderyObjectEventData Class

The response data for bindery events is returned as a BinderyObjectEventData class. This class consists of the read-only properties as explained in the table below.

Table 4-22 *Details of the BinderyObjectEventData Class*

Property	Type	Description
entry	LDAPDN	Specifies the DN of the Directory entry that is being created to represent the bindery object.
type	INTEGER	Specifies the bindery object type.
emuObjFlags	INTEGER	Specifies the bindery object flags.
security	INTEGER	Specifies the bindery object security.
name	LDAPString	Specifies the name of the bindery object.

4.3.6 SecurityEquivalence Event

This section details the SecurityEquivalence event.

SecurityEquivalenceEventData Class

The response data for SecurityEquivalence events is returned as a SecurityEquivalenceEventData class. This class consists of the read-only properties as explained in the table below.

Table 4-23 *Details of the SecurityEquivalenceEventData Class*

Property	Type	Description
entry	LDAPDN	Specifies the DN of the Directory object whose Security Equivalence Vector (SEV) is being checked.
retryCount	INTEGER	Specifies the number of retries.
valueDN	LDAPDN	Specifies the DN of an object or group being checked.
referral	List of ReferralAddress	Specifies the List of referrals.
referralCount	INTEGER	Specifies the number of referrals in the referrals parameter.

4.3.7 Module State Events

This section details the module state events.

ModuleStateEventData Class

The response data for module state events is returned as a ModuleStateEventData class. This class consists of the read-only properties explained in the table below.

Table 4-24 *Details of the Read-only Properties*

Property	Type	Design
ConnectionDN	LDAPDN	Specifies the DN of the entry associated with the connection.
Flags	INTEGER	The least significant byte of the flags field contains module attribute flags. The next byte contains event subtype flags. They indicate the type of module event in progress. See the Table below for details.
Name	LDAPSTRING	Specifies the affected module name.
Description	LDAPSTRING	Specifies the name and description of the target module.
Source	LDAPSTRING	Specifies the affecting module.

The values for flags field are contained in the following table:

Table 4-25 *Values for Flags Field*

0x0001	DSE_MOD_HIDDEN
0x0002	DSE_MOD_SYSTEM
0x0004	DSE_MOD_ENGINE
0x0008	DSE_MOD_AUTOMATIC
0x00FF	DSE_MOD_FILE_MASK
0x0100	DSE_MOD_POSTEVENT
0x0200	DSE_MOD_AVAILABLE
0x0400	DSE_MOD_LOADING
0x0800	DSE_MOD_MODIFY
0x8000	DSE_MOD_NEGATE_BIT
0xFF00	DSE_MOD_EVENT_MASK

4.3.8 Network Address Event

This section details the network address event.

NetworkAddressEventData Class

The response data for network address events is returned as a NetworkAddressEventData class. This class consists of the read-only properties as explained in the table below.

Table 4-26 *Details of the NetworkAddressEventData Class*

Property	Type	Description
type	INTEGER	Specifies the type of the address. Can be one of the following values: ◆ NT_IPX ◆ NT_IP ◆ NT_SDLC ◆ NT_TOKENRING_ETHERNET ◆ NT_OSI ◆ NT_APPLETALK ◆ NT_COUNT
data	OCTET STRING	A character array containing address.

Remarks

The address is stored as a binary string. This string is the literal value of the address. To display as a hexadecimal value, you must convert each 4-bit nibble to the correct character (0,1,2,3,...) For two net addresses to match, the type, length, and value of the addresses must match.

4.3.9 Connection Change Event

This section explains the properties of the ConnectionStateEventData class and flags.

ConnectionStateEventData Class

The response data for connection change events is returned as a ConnectionStateEventData class. This class consists of the read-only properties as explained in the table below.

Table 4-27 *Read-Only Properties*

Property	Type	Design
ConnectionDN	LDAPDN	Specifies the DN of the entry associated with the connection.
OldFlags	INTEGER	Specifies the flag associated with the previous connection state, and is one of the flag Specified in Table below.
NewFlags	INTEGER	Specifies the flag that indicates the new connection state. Uses the same flags as oldFlags.
SourceModule	LDAPSTRING	Specifies the module that caused the connection state to change.

Flags

Table 4-28 *Values for Flag Field*

0x00000001	DSE_CONN_VALID
0x00000002	DSE_CONN_AUTHENTIC
0x00000004	DSE_CONN_SUPERVISOR
0x00000008	DSE_CONN_OPERATOR
0x00000010	DSE_CONN_LICENSED
0x00000020	DSE_CONN_SEV_IS_STALE
0x000000FF	DSE_CONN_OPERATIONAL_FLAGS
0x00010000	DSE_CONN_CLEAR_ON_UNLOCK
0x00020000	DSE_CONN_LOCKED
0x00040000	DSE_CONN_CLEAR_ON_EVENT
0x000F0000	DSE_CONN_SECURITY_FLAGS

5 LDAP Default DSML Serialization

LDAP Directory Services Markup Language (DSML) serialization provides a way for applications to store the state of LDAP runtime objects (that is, an object of class LDAPEntry) in the XML format pertaining to the DSML schema constraints.

The stored destination may be any of the following:

- Runtime output stream
- User And/or application
- Console and/or terminal
- A permanent file in OS file system

The stored objects can be read and later constructed depending on the application use and demand.

5.1 Concepts

The runtime object retrieved is exactly what it was used to be before storing the state to a file. This is a customized implementation on serialization protocol supported in Java. In Java serialization, the output format of the runtime stored objects is binary. Where as, the DSML serialization supported in the LDAP classes allows the objects to be stored in the XML format. The resultant XML is almost more than 95% application data (except the class header and some serialization protocol specific demarcation bytes embedded inside the application data, which has to be there as specified in the serialization protocol.) Hence, it is easier to visualize the data in addition to the benefits provided by the Java serialization protocol. The reason why it is called as default serialization is because the client applications can use the same set of API calls as provided by Java serialization to achieve the functionality provided by Java serialization protocol and yet achieve a friendly XML output structure or file.

5.1.1 Advantages of DSML Serialization

- The output format is in XML in accordance with the DSML schema. Therefore, it facilitates an LDAP administrator to better administration and visualization of stored data.
- Version compatibility. Supports reading deprecated versions of LDAP objects.
- Provides same interface to the client applications as provided by Java serialization protocol.
- Provides a backup and restore service.

The LDAP Libraries for Java (http://developer.novell.com/ndk/jldap.htm) are enhanced to provide support for DSML Serialization.

5.2 Classes

This section lists the classes that support default DSML serialization.

LDAP attribute, LDAP attribute set, LDAP entry and related classes

- LDAPAttribute.java

- LDAPAttributeSet.java
- LDAPEntry.java

LDAP schema and schema element classes

- LDAPAttributeSchema.java
- LDAPDITContentRuleSchema.java
- LDAPDITStructureRuleSchema.java
- LDAPMatchingRuleSchema.java
- LDAPMatchingRuleUseSchema.java
- LDAPNameFormSchema.java
- LDAPObjectClassSchema.java
- LDAPSchema.java
- LDAPSyntaxSchema.java

LDAP control and related classes

- LDAPControl.java

LDAP message and related classes

- LDAPAddRequest.java
- LDAPCompareRequest.java
- LDAPDeleteRequest.java
- LDAPExtendedOperation.java
- LDAPExtendedRequest.java
- LDAPExtendedResponse.java
- LDAPMessage.java
- LDAPModification.java
- LDAPModifyDNRequest.java
- LDAPModifyRequest.java
- LDAPResponse.java
- LDAPSearchRequest.java
- LDAPSearchResult.java
- LDAPSearchResultReference.java

LDAP URL and related classes

- LDAPUrl.java

Other classes

- LDAPSearchResults.java

5.3 Using the Application Client

This section provides steps for storing and/or retrieving the state of an LDAP object by an application client. The examples in the procedure below demonstrates storing and/or retrieving an object of the LDAPAttribute class.

5.3.1 Serializing an Object

1. Specify the -s option to serialize an object.
2. Specify the host name of Directory server.
3. Specify login DN and password of the user or administrator.
4. Specify the LDAP entry and attribute name of the entry you want to store and/or serialize.
5. Specify the destination where you want to store the attribute object (in this case it is to a file in the OS). The specified attribute object is read from the LDAP server.
6. Pass the LDAP attribute object read from the directory to the serialization API as follows:

```
//call serialization method
      try{
      //Construct an Output stream to a File
    ObjectOutputStream out =
                  new ObjectOutputStream(
                                    new
FileOutputStream(fileName));
      //pass the read LDAP attribute object as an argument
    //call writeObject method to write the object
    out.writeObject(ldapattr);

   //close the stream after object is written
            out.close();

   //print the serialized object
   System.out.println("Object written =" + ldapattr.toString());
      }
      //Handle any exception occurred
      catch(IOException e){
         System.out.println("Error: " + e.toString());
}
```

After entering the above, the XML structure written to the selected file is similar to the following:

```
¬í _sr _com.novell.ldap.LDAPAttributeÖÌ_Fí
*******************************************************************
*****
** The encrypted data above and below is the Class definition and
***
** other data specific to Java Serialization Protocol. The data
*******
** which is of most application specific interest is as follows...
*****
*******************************************************************
******
****************** Start of application
```

```
data**************************
*******************************************************************
******
<LDAPAttribute>
    <attr name="uniqueID">
        <value>user10</value>
    </attr>
</LDAPAttribute>
*******************************************************************
******
****************** End of application
data***************************
*******************************************************************
******
x
```

In the above output, the data below and above the start and end of application data headings respectively, is of application-specific interest. Other data above and below these headings are specific to the serialization protocol. Also, some serialization-specific demarcation bytes are embedded in the actual data if the data stored is large. An administrator can easily visualize the data presented in this block. If the data is presented in purely raw bytes as obtained by using Java serialization only, it would be very difficult for a user to perceive. This is where the default DSML serialization provides added service over Java serialization.

5.3.2 De-Serializing an Object

1. Specify the -d option to de-serialize an object.
2. Specify the filename from which the LDAPAttribute object is to be read.
3. Call the de-serialization API to read and construct the LDAPAttribute object as follows:

```
//call de-serialization method
      try{
   //Construct an Input stream to File
            ObjectInputStream in =
                  new ObjectInputStream(
                        new FileInputStream(fileName));

   //call readObject method to read and construct the LDAP
attribute object
   LDAPAttribute ldapattr = (LDAPAttribute)in.readObject();

   //Print the runtime object after de-serialization
   System.out.println("Object read =" + ldapattr.toString());

   //close the input stream
   in.close();
      }
      //Handle any exceptions occurred
catch(IOException io){
            System.out.println("Error: " + io.toString());
      }
      catch(ClassNotFoundException ce){
```

```
                    System.out.println("Error: " + ce.toString());
        }
```

To view and execute the complete code, refer to the following sample programs in LDAP SDK:

- LDAPAttributeDSMLSerialization.java
- LDAPSchemaDSMLSerialization.java

5.4 Supplements to Default DSML Serialization

LDAP Libraries for Java also supports APIs that write and read runtime objects of LDAP classes, although this is strictly not using the Java serialization protocol neither customizing its implementation.

This support is an alternative to the support provided by java.beans.XMLEncoder or XMLDecoder in Java. In fact our support is one step ahead as provided by Java XMLEncoder or XMLDecoder. Java XMLEncoder or XMLDecoder stores XML data that pertain to some schema (may be Java Bean specific schema) which is not needed for LDAP classes.

Our implementation stores the data in complete (100% XML structure with no class header and serialization demarcation bytes) DSML format.

The following two methods support this feature in LDAP Libraries for Java as listed in Section 5.2, "Classes," on page 69:

- `public void writeDSML (java.io.OutputStream out) throws java.io.IOException`
- `public static Object readDSML (InputStream input) throws java.io.IOException`

Client applications need to call these methods explicitly for storing the data and read from streams.

LDAP Tools

6

A number of tools have been developed to import entries from a file to an LDAP directory, to modify the entries in a directory from a file, and to export the entries to a file. These tools also support schema modifications by adding attribute and class definitions from a file.

These tools are included in the LDAP Libraries for C download, and they are dependent upon those libraries to function. Documentation for these tools are included with that component.

For more information on LDAP tools, see the LDAP Tools documentation, or the LDAP Libraries for C download (http://developer.novell.com/ndk/cldap.htm).

JavaDoc API Reference

7

com.novell.ldap Javadoc for the com.novell.ldap package is located on the Web (../api/index.html), or on the local disk once the documentation has been installed (default location is C:\Novell\NDK\jldap\doc\jldapenu\api)

org.ietf.ldap Javadoc for the org.ietf.ldap package is located on the Web (../ietfapi/index.html), or on the local disk once the documentation has been installed (default location is C:\Novell\NDK\jldap\doc\jldapenu\ietfapi)

For the differences between the org.ietf.ldap and the com.novell.ldap packages see "LDAP Classes" on page 10.

The LDAP Classes for Java, eDirectory Technical Reference and the LDAP and eDirectory Integration manuals are also available on the Web (http://developer.novell.com/ndk/doc_jldap.htm), or on the local disk once they have been installed (default locations are C:\Novell\NDK\doc\ndslib and C:\Novell\NDK\doc\ldapover).

Revision History

A

The following table lists all changes made to the LDAP Classes for Java documentation:

February 2008	◆ Added two controls to the Section 3.1, “Supported Controls,” on page 35. ◆ Added the functionality of Paged Results Control to the Section 3.1, “Supported Controls,” on page 35
October 2007	Added two new LDAP extensions to the Section 3.2, “Extensions,” on page 36
June 2006	Extended Constructor information to the Section 1.3.1, “LDAP Connections,” on page 14.
March 2006	Fixed formatting issues. Added information about LDAPConnection constructor in the Section 1.3.1, “LDAP Connections,” on page 14.
June 2005	Added the following: ◆ “SCOPE_SUBORDINATESUBTREE” on page 18. ◆ “Other classes” on page 70.
March 2005	◆ Added information on Object based backup/restore extensions in the LDAP Classes for Java Section 3.2, “Extensions,” on page 36. For more information on Extensions, refer Javadoc APIs (http://developer.novell.com/ndk/doc/jldap/jldapenu/api/index.html). ◆ Removed the link to ldapzone from Section 1.7, “Other Sources of LDAP and eDirectory Information,” on page 28.
October 2004	Added information about Chapter 5, “LDAP Default DSML Serialization,” on page 69.
June 2004	Added information on the following classes: ◆ “EntryEventData Class” on page 60 ◆ “DSETimestamp Class” on page 60 ◆ “ValueEventData Class” on page 61 ◆ “DebugEventData Class” on page 61 ◆ “DebugParameter Class” on page 62 ◆ “ReferralAddress Class” on page 63 ◆ “GeneralDSEventData Class” on page 64 ◆ “BinderyObjectEventData Class” on page 64 ◆ “SecurityEquivalenceEventData Class” on page 65 ◆ “ModuleStateEventData Class” on page 65 ◆ “NetworkAddressEventData Class” on page 66 ◆ “ConnectionStateEventData Class” on page 67
February 2003	Added information on the new LDAP extensions supported.

October 2003	Added the following: ◆ HP-UX support. ◆ SASL bind methods implemented.
March 2003	Added information on using the debug version of the classes.
September 2002	Added information on referral handling for non-search operations and updated javadoc to reflect recent draft updates. See the readme for details.
May 2002	Updated JDK requirements
February 2002	Updated SSL integration instructions
September 2001	Expanded and revised the searching and referral sections, and added information about the org.ietf.ldap package
June 2001	Added the following: ◆ Updated information on referral and error handling, LDAP URLs and LDAP Messages. ◆ Added schema read samples. ◆ Added information on LDAP controls and extensions.
February 2001	Added as a new component on the NDK.