Novell Developer Kit

www.novell.com

LDAP CLASSES FOR JAVA*

October 2003

Novell®

Legal Notices

Novell, Inc.
1800 South Novell Place
Provo, UT 84606
U.S.A.

www.novell.com

LDAP Classes for Java
October 2003

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.

Novell Trademarks

AppNotes is a registered trademark of Novell, Inc.
AppTester is a registered trademark of Novell, Inc., in the United States.
ASM is a trademark of Novell, Inc.
BorderManager is a registered trademark of Novell, Inc.
BrainShare is a registered service mark of Novell, Inc., in the United States and other countries.
C3PO is a trademark of Novell, Inc.
Client 32 is a trademark of Novell, Inc.
ConsoleOne is a registered trademark of Novell, Inc.
Controlled Access Printer is a trademark of Novell, Inc.
Custom 3rd-Party Object is a trademark of Novell, Inc.
DeveloperNet is a registered trademark of Novell, Inc. in the United States and other countries.
DeveloperNet 2000 is a trademark of Novell, Inc.
Direct Connect is a service mark of Novell, Inc.
DirXML is a registered trademark of Novell, Inc.
eDirectory is a trademark of Novell, Inc.
Full Service Directory is a trademark of Novell, Inc.
GroupWise is a registered trademark of Novell, Inc. in the United States and other countries.
Hardware Specific Module is a trademark of Novell, Inc.
Hot Fix is a trademark of Novell, Inc.
iChain is a registered trademark of Novell, Inc.
Internetwork Packet Exchange is a trademark of Novell, Inc.
IPX is a trademark of Novell, Inc.
IPX/SPX is a trademark of Novell, Inc.
Link Support Layer is a trademark of Novell, Inc.
LSL is a trademark of Novell, Inc.
ManageWise is a registered trademark of Novell, Inc., in the United States and other countries.
Mirrored Server Link is a trademark of Novell, Inc.
MSL is a trademark of Novell, Inc.
MSM is a trademark of Novell, Inc.
NCP is a trademark of Novell, Inc.
NDPS is a registered trademark of Novell, Inc.
NDS is a registered trademark of Novell, Inc. in the United States and other countries.
NDS Administrator is a trademark of Novell, Inc.
NDS Manager is a trademark of Novell, Inc.
NE2000 is a trademark of Novell, Inc.
NetWare is a registered trademark of Novell, Inc. in the United States and other countries.
NetWare/IP is a trademark of Novell, Inc.
NetWare Connect is a registered trademark of Novell, Inc. in the United States.
NetWare Core Protocol is a trademark of Novell, Inc.
NetWare Loadable Module is a trademark of Novell, Inc.
NetWare Management Portal is a trademark of Novell, Inc.
NetWare Management System is a trademark of Novell, Inc.
NetWare MHS is a registered trademark of Novell, Inc.
NetWare Name Service is a trademark of Novell, Inc.
NetWare Peripheral Architecture is a trademark of Novell, Inc.
NetWare Print Server is a trademark of Novell, Inc.
NetWare Requester is a trademark of Novell, Inc.
NetWare SFT and NetWare SFT III are trademarks of Novell, Inc.
NetWare SQL is a trademark of Novell, Inc.
NetWire is a registered service mark of Novell, Inc. in the United States and other countries.
NIMS is a trademark of Novell, Inc.
NLM is a trademark of Novell, Inc.
NMAS is a trademark of Novell, Inc.
NMS is a trademark of Novell, Inc.

Novell is a registered trademark of Novell, Inc. in the United States and other countries.
Novell Authorized Reseller is a service mark of Novell, Inc.
Novell Authorized Service Center is a service mark of Novell, Inc.
Novell Certificate Server is a trademark of Novell, Inc.
Novell Client is a trademark of Novell, Inc.
Novell Cluster Services is a trademark of Novell, Inc.
Novell Directory Services is a registered trademark of Novell, Inc.
Novell Distributed Print Services is a trademark of Novell, Inc.
Novell Internet Messaging System is a trademark of Novell, Inc.
Novell Labs is a trademark of Novell, Inc.
Novell Press is a trademark of Novell, Inc.
Novell SecretStore is a trademark of Novell, Inc.
Novell Security Attributes is a trademark of Novell, Inc.
Novell Storage Services is a trademark of Novell, Inc.
Novell Web Server is a trademark of Novell, Inc.
Novell, Yes, Tested & Approved logo is a trademark of Novell, Inc.
NSI is a trademark of Novell, Inc.
ODI is a trademark of Novell, Inc.
Open Data-Link Interface is a trademark of Novell, Inc.
Packet Burst is a trademark of Novell, Inc.
Printer Agent is a trademark of Novell, Inc.
QuickFinder is a trademark of Novell, Inc.
Red Box is a trademark of Novell, Inc.
Remote Console is a trademark of Novell, Inc.
RX-Net is a trademark of Novell, Inc.
Sequenced Packet Exchange is a trademark of Novell, Inc.
SFT and SFT III are trademarks of Novell, Inc.
SPX is a trademark of Novell, Inc.
Storage Management Services is a trademark of Novell, Inc.
System V is a trademark of Novell, Inc.
Topology Specific Module is a trademark of Novell, Inc.
Transaction Tracking System is a trademark of Novell, Inc.
TSM is a trademark of Novell, Inc.
TTS is a trademark of Novell, Inc.
Universal Component System is a registered trademark of Novell, Inc.
Virtual Loadable Module is a trademark of Novell, Inc.
VLM is a trademark of Novell, Inc.
ZENworks is a registered trademark of Novell, Inc.

Third-Party Trademarks

All third-party trademarks are the property of their respective owners.

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 document explains how to use these libraries to access Novell eDirectory. However, they can be used to access any LDAP-enabled directory.

1 Concepts

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

- *NDS Technical Overview*
- *LDAP and NDS*

These manuals are available on the Web (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).

Getting Started

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

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 applicaton. (The classes no longer support JDK 1.1.7).

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 "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).

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 RedHat 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 RedHat 6.2, and 7.2)
- AIX 4.3
- HP-UX 11.11

LDAP Classes

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

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 is 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 “Integrating SSL with the LDAP Classes” on page 12

Sample Code

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

LDAP Test Server

Novell has set up an LDAP server that you can access over the Internet to test your LDAP application. The server's name is www.nldap.com, and it listens on the default LDAP port (389). To use authenticated access, you must set up your own account in your own eDirectory container. Your account is limited to 1MB of disk storage.

To access this site, go to the NDS/LDAP Services Access Test Site (http://www.nldap.com/NLDAP).

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:

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

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

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 1* (http://developer.novell.com/ndk/doc/ncslib/pkis_enu/data/h1axgumd.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 NDS" in the November 1998 issue of Novell Developer Notes (http://developer.novell.com/research/devnotes/1998/november/a1frame.htm).

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 folowing:

HINT: 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. Create an ssl.properties file and include the following lines:

   ```
   ssl.keystore = <keystore filename>
   ssl.keystore.password = <password>
   ```

   Replace <password> with the password for the KeyStore file. Using the keystore created in the example above, the keystore filename would be sslkey.keystore.
4. 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());
```

5. 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.

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.

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.

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 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 39.

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.

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.

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).

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 39.

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 will apply 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 will retrieve 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);
```

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 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 39.

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 24.

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.

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:

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 vaild starting point.

Search Scope

The search scope parameter specifies the depth of the search and can be one of three 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.

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 emailaddress. 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 atributes and their values).

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 will return 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);
```

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 ) {
```

Once 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.

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 2 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 3 LDAP Filter Boolean Operators

Boolean Operators	Description
&	And. For example, (&(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.

Filter and Description
(&(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.

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 NDS Integration Guide*. The LDAP servers in releases previous to 8.5 do not support requesting operational attributes in a search operation.

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.

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 will allow you to enumerate through the results without ever blocking.

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());
  }
}
```

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 “Referral Handling in LDAP v3” on page 25.

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.

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).

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)

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.

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.

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.

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.

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.

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/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 NDS." 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.
- Novell’s eDirectory/LDAP Services Access Test Site (http://nldap.com/nldap). This site contains links to LDAP RFCs and access to a Novell LDAP server for testing your LDAP application.
- LDAPZone.com (http://www.ldapzone.com). Contains LDAP-related forums, information, and articles.
- “Netscape Directory SDK 3.0 for Java Programmer's Guide (http://developer.netscape.com/docs/manuals/directory.html).” 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 chapter 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.

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 “Integrating SSL with the LDAP Classes” on page 12.

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.

Once 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.

Modifying an Entry

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

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.

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.

To read the Root DSE complete the following four 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 will return one entry which will be the root DSE.

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.

To read the schema complete the following four steps:

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

   NOTE: 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 the Chapter 5, “JavaDoc API Reference,” on page 39, 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 NDS and LDAP Integration 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.

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

- **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.

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.

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
2.16.840.1.113719.1.27.100.29	AbortPartitionOperationRequest
2.16.840.1.113719.1.27.100.30	AbortPartitionOperationResponse

OID	Name
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

4 LDAP Tools

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).

5 JavaDoc API Reference

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\doc\jldap\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\doc\jldap\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, NDS Technical Reference and the LDAP and NDS 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

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

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.