Network Working Group Rob Weltman
INTERNET-DRAFT Netscape Communications Corp.
Tim Howes
Netscape Communications Corp.
Mark Smith
Netscape Communications Corp.
September 26, 1997

The Java LDAP Application Program Interface
draft-ietf-asid-ldap-java-api-01.txt

Status of this Memo

Abstract

This document defines a java language application program interface to the lightweight directory access protocol (LDAP), in the form of a class library. It complements but does not replace RFC 1823 ([9]), which describes a C language application program interface. It updates and replaces draft-ietf-asid-ldap-java-api-00.txt [13], in adding clarification on behavior under various circumstances, and in revising support for language-sensitive attribute parsing. Other additions and corrections are listed in the appendix.

1. Introduction

The LDAP class library is designed to provide powerful, yet simple, access to a LDAP directory services. It defines a synchronous interface to LDAP, with support for partial results on searching, to suit a wide variety of applications. This document gives a brief overview of the

LDAP model, then an overview of the constituents of the class library. The public class methods are described in detail, followed by an appendix that provides some example code demonstrating the use of the classes, and an appendix listing changes from earlier drafts.

2. Overview of the LDAP model

LDAP is the lightweight directory access protocol, described in [2] and [7]. It defines a lightweight access mechanism in which clients send requests to and receive responses from LDAP servers.

The LDAP information model comes from X.500 [1] and is based on the entry, which contains information about some object (e.g., a person). Entries are composed of attributes, which have a type and one or more values. Each attribute has a syntax that determines what kinds of values are allowed in the attribute (e.g., ASCII characters, a jpeg photograph, etc.) and how those values behave during directory operations (e.g., is case significant during comparisons).

Entries may be organized in a tree structure, usually based on political, geographical, and organizational boundaries. Other structures are possible, including a flat namespace. Each entry is uniquely named relative to its sibling entries by its relative distinguished name (RDN) consisting of one or more distinguished attribute values from the entry. At most one value from each attribute may be used in the RDN. For example, the entry for the person Babs Jensen might be named with the "Barbara Jensen" value from the commonName attribute.

A globally unique name for an entry, called a distinguished name or DN, is constructed by concatenating the sequence of RDNs from the entry up to the root of the tree. For example, if Babs worked for the University of Michigan, the DN of her U-M entry might be "cn=Barbara Jensen, o=University of Michigan, c=US". The DN format used by LDAP is defined in [4].

Operations are provided to authenticate, search for and retrieve information, modify information, and add and delete entries from the tree.

An LDAP server may return referrals if it cannot completely service a request (for example if the request specifies a directory base outside of the tree managed by the server). The LDAP class library offers the programmer two options: the programmer can catch these referrals as exceptions and explicitly issue new requests to the referred-to servers, or the programmer can let the library automatically follow the referrals. In the latter case, the programmer may also provide a reauthentication object, allowing automatic referrals to proceed with appropriate credentials (as opposed to anonymous authentication) for each referred-to server.

Before the client encodes and sends a string value to a server, the string values are converted from the java 16-bit Unicode format to UTF-8 format. Then, when the values are received by the server, the values are converted back to 16-bit Unicodeformat. The integrity of double-byte and other non-ASCII character sets is fully preserved.

The next sections give an overview of how the class library is used and detailed descriptions of the LDAP class methods that implement all of these functions.

3. Overview of the LDAP classes

The central LDAP class is LDAPConnection. It 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 LDAPConnection class also provides fields for storing settings that are specific to the LDAP session (such as limits on the number of results returned or timeout limits). An LDAPConnection object can be cloned, allowing objects to share a single network connection but use different settings (using LDAPConnection.setOption()).

To support extensions of the LDAP protocol LDAPConnection and to add the capability to determine the LDAP protocol level supported by an LDAPConnection object, LDAPConnection implements a protocol interface, which is currently LDAPv3.

A search conducted by an LDAPConnection object returns results in an LDAPSearchResults object, which can be enumerated to access the entries found. Each entry (represented by an LDAPEntry object) provides access to the attributes (represented by LDAPAttribute objects) returned for that entry. Each attribute can produce the values found as byte arrays or as Strings.

3.1. Interfaces

LDAPEntryComparator	An interface to support arbitrary sorting algorithms for entries returned by a search operation. The basic java LDAP classes include one implementation: LDAPCompareAttrNames, to sort in ascending order based on one or more attribute names.
LDAPRebind	A programmer desiring reauthentication on automatically following referrals must implement this

	interface. Without it, automatically followed referrals will use anonymous authentication. Referrals of any type other to an LDAP server (i.e. a referral URL other than ldap://something) are ignored on automatic referral following.
LDAPSocketFactory	Programmers needing to provide or use specialized socket connections, including Transport Layer Security (TLS) based ones, can provide an object constructor to implement them using this interface.
LDAPv2	This interface defines the functionality of the LDAP version 2 protocol. It is implemented by LDAPConnection.
LDAPv3	This interface extends LDAPv2, adding the functionality of the LDAP version 3 protocol. It is implemented by LDAPConnection.

3.2. Classes

LDAPAttribute	Represents the name and values of one attribute of a directory entry.
LDAPAttributeSet	A collection of LDAPAttributes.
LDAPCompareAttrNames	An implementation of LDAPEntryComparator, to support sorting of search results by one or more attributes.
LDAPConnection	Implements LDAPv3 and is the central point for interactions with a directory.
LDAPControl	Sets additional parameters or constraints for an LDAP operation, either on the server or on the client.

LDAPDN	A utility class to facilitate composition and decomposition of distinguished names (DNs).
LDAPEntry	Represents a single entry in a directory.
LDAPExtendedOperation	Encapsulates the ID and data associated with the sending or receiving of an extended operation (an LDAPv3 feature).
LDAPModification	A single add/delete/replace operation to an LDAPAttribute.
LDAPModificationSet	A collection of LDAPModifications
LDAPRebindAuth	An implementation of LDAPRebind must be able to provide an LDAPRebindAuth at the time of a referral. The class encapsulates reauthentication credentials.
LDAPSearchConstraints	Defines the options controlling search operations.
LDAPSearchResults	The enumerable results of a search operation.
LDAPSortKey	Specifies how search results are to be sorted.
LDAPUrl	Encapsulates parameters of an LDAP Url query, as defined in [8].

3.3. Exceptions

LDAPException	General exception, which includes an error message and an LDAP error code.
LDAPReferralException	Derived from LDAPException and contains a list of LDAPUrls corresponding to referrals received on an LDAP operation.

4. Overview of LDAP API use

An application generally uses the LDAP API in four steps.

- Construct an LDAPConnection. Initialize an LDAP session with a directory server. The LDAPConnection.connect() call establishes a handle to the session, allowing multiple sessions to be open at once, on different instances of LDAPConnection.

- Authenticate to the LDAP server with LDAPConnection.authenticate().

- Perform some LDAP operations and obtain some results. LDAPConnection.search() returns an LDAPSearchResults, which can be enumerated to access all entries found. LDAPConnection.read() returns a single entry.

- Close the connection. The LDAPConnection.disconnect() call closes the connection.

All operations are synchronous - they do not return until the operation has completed. In the java environment, it is appropriate to create a thread for the operation rather than providing parallel synchronous and asynchronous operations, as is the case in the C language API described in [9]. To facilitate user feedback during 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.

Standard java Enumerations are used to parse search results. Errors result in the throwing of an LDAPException, with a specific error code and context-specific textual information available.

The following sections describe the LDAP classes in more detail.

5. The java LDAP classes

5.1. public class LDAPAttribute

The LDAPAttribute class represents the name and values of an attribute. It is used to specify an attribute to be added to, deleted from, or modified in a Directory entry. It is also returned on a search of a Directory.

5.1.1. Constructors

```
   public LDAPAttribute(String attrName)
```

Constructs an attribute with no values.

```
public LDAPAttribute(String attrName,
                     byte attrBytes[])
```

Constructs an attribute with a byte-formatted value.

```
public LDAPAttribute(String attrName,
                     String attrString)
```

Constructs an attribute that has a single string value.

```
public LDAPAttribute(String attrName,
                     String attrStrings[])
```

Constructs an attribute that has an array of string values.

Parameters are:

attrName	Name of the attribute.
attrBytes	Value of the attribute as raw bytes.
attrString	Value of the attribute as a String.
attrStrings	Array of values as Strings.
attrBytesArray	Array of values as raw byte arrays.

5.1.2. addValue

```
public synchronized void addValue(String attrString)
```

Adds a string value to the attribute.

```
public synchronized void addValue(byte attrBytes[])
```

Adds a byte[]-formatted value to the attribute.

Parameters are:

attrString	Value of the attribute as a String.

attrBytes	Value of the attribute as raw bytes.

5.1.3. getByteValues

```
public Enumeration getByteValues()
```

Returns an enumerator for the values of the attribute in byte[] format.

5.1.4. getStringValues

```
public Enumeration getStringValues()
```

Returns an enumerator for the string values of an attribute.

5.1.5. getName

```
public String getName()
```

Returns the name of the attribute.

5.1.6. removeValue

```
public synchronized void removeValue(String attrString)
```

Removes a string value from the attribute.

```
public synchronized void removeValue(byte attrValue[])
```

Removes a byte[]-formatted value from the attribute.

Parameters are:

attrString	Value of the attribute as a String.
attrBytes	Value of the attribute as raw bytes.

5.1.7. size

```
public int size()
```

Returns the number of values of the attribute.

5.2. public class LDAPAttributeSet

An LDAPAttributeSet is a collection of LDAPAttributes, as returned in an entry on a search or read operation, or used to construct an entry to be added to a directory.

5.2.1. Constructors

```
public LDAPAttributeSet()
```

Constructs a new set of attributes. This set is initially empty.

5.2.2. add

```
public synchronized void add(LDAPAttribute attr)
```

Adds the specified attribute to this attribute set.

Parameters are:

attr	Attribute to add to this set.

5.2.3. elementAt

```
public LDAPAttribute elementAt(int index)
                            throws ArrayIndexOutOfBoundsException
```

Returns the attribute at the position specified by the index. The index is 0-based.

Parameters are:

index	Index of the attribute to get.

5.2.4. getAttribute

```
public LDAPAttribute[] getAttribute(String attrName)
```

Returns the attribute matching the specified attrName. For example, getAttribute("cn") returns only the "cn" attribute; getAttribute("cn;lang-en") returns only the "cn;lang-en" attribute. In both cases, null is returned if there is no exact match to the specified attrName.

```
public LDAPAttribute[] getAttribute(String attrName, String lang)
```

Returns a single best-match attribute, or none if no match is available in the entry.

LDAP version 3 allows adding a subtype specification to an attribute name. "cn;lang-ja", for example, indicates a Japanese language subtype of the "cn" attribute. "cn;lang-ja-JP-kanji" may be a subtype of "cn;lang-ja". This feature may be used to provide multiple localizations in the same Directory. For attributes which do not vary among localizations, only the base attribute may be stored, whereas for others there may be varying degrees of specialization.

getAttribute(attrName,lang) returns the subtype that matches attrName and that best matches lang. If there are subtypes other than "lang" subtypes included in attrName, e.g. "cn;binary", only attributes with all of those subtypes are returned. If lang is null or empty, the method behaves as getAttribute(attrName). If there are no matching attributes, null is returned.

```
Example:
   Assume the entry contains only the following attributes:
      cn;lang-en
      cn;lang-ja-JP-kanji
      sn
   getAttribute( "cn" ) returns null.
   getAttribute( "sn" ) returns the "sn" attribute.
   getAttribute( "cn", "lang-en-us" ) returns the "cn;lang-en" attribute.
   getAttribute( "cn", "lang-en" ) returns the "cn;lang-en" attribute.
   getAttribute( "cn", "lang-ja" ) returns null.
   getAttribute( "sn", "lang-en" ) returns the "sn" attribute.
```

Parameters are:

attrName	The name of an attribute to retrieve, with or without subtype specification(s). "cn", "cn;phonetic", and "cn;binary" are valid attribute names.
lang	A language specification as in [14], with optional subtypes appended using "-" as separator. "lang-en", "lang-en-us", "lang-ja", and "lang-ja-JP-kanji" are valid language specification.

5.2.5. getAttributes

```
public Enumeration getAttributes()
```

Returns an enumeration of the attributes in this attribute set.

5.2.6. remove

```
public synchronized void remove(String name)
```

Removes the specified attribute from the set. If the attribute is not a member of the set, nothing happens.

Parameters are:

name	Name of the attribute to remove from this set. To remove an LDAPAttribute by object, the LDAPAttribute.getName method can be used:

```
LDAPAttributeSet.remove( attr.getName() );
```

5.2.7. removeElementAt

```
public void removeElementAt(int index)
                            throws ArrayIndexOutOfBoundsException
```

Removes the attribute at the position specified by the index. The index is 0-based.

Parameters are:

index	Index of the attribute to remove.

5.2.8. size

```
public int size()
```

Returns the number of attributes in this set.

5.3. public class LDAPCompareAttrNames implements LDAPEntryComparator

An object of this class supports sorting search results by attribute name, in ascending or descending order.

5.3.1. Constructors

```
public LDAPCompareAttrNames(String attrName)
```

Constructs an object that will sort results by a single attribute, in ascending order.

```
public LDAPCompareAttrNames(String attrName,
                            boolean ascendingFlag)
```

Constructs an object that will sort results by a single attribute, in either ascending or descending order.

```
public LDAPCompareAttrNames(String[] attrNames)
```

Constructs an object that will sort by one or more attributes, in the order provided, in ascending order.

```
public LDAPCompareAttrNames(String[] attrNames,
                            boolean[] ascendingFlags)
                            throws LDAPException
```

Constructs an object that will sort by one or more attributes, in the order provided, in either ascending or descending order for each attribute.

Parameters are:

attrName	Name of an attribute to sort by.
attrNames	Array of names of attributes to sort by.
ascendingFlag	true to sort in ascending order, false for descending order.
ascendingFlags	Array of flags, one for each attrName, where each one is true to sort in ascending order, false for descending order. An LDAPException is thrown if the length of ascendingFlags is not greater than or equal to the length of attrNames.

5.3.2. isGreater

```
public boolean isGreater (LDAPEntry entry1, LDAPEntry entry2)
```

Returns true if entry1 is to be considered greater than entry2, for the purpose of sorting, based on the attribute name or names provided on object construction.

Parameters are:

entry1	Target entry for comparison.
entry2	Entry to be compared to.

5.4. public class LDAPConnection implements LDAPv3, Cloneable

LDAPConnection is the central class that encapsulates the connection to a Directory Server through the LDAP protocol. It implements the LDAPv2 and LDAPv3 interfaces. An LDAPConnection object is not connected on construction, and may only be connected to one server at one port. Multiple threads may share this single connection, and an application may have more than one LDAPConnection object, connected to the same or different Directory Servers.

Besides the methods described for LDAPv2 and LDAPv3, LDAPConnection provides the following methods.

5.4.1. Constructors

public LDAPConnection()

Constructs a new LDAPConnection object, which represents a connection to an LDAP server.

Calling the constructor does not actually establish the connection. To connect to the LDAP server, use the connect method.

public LDAPConnection(LDAPSocketFactory factory)

Constructs a new LDAPConnection object, which will use the supplied class factory to construct a socket connection during LDAPConnection.connect().

Parameters are:

factory	An object capable of producing a Socket.

5.4.2. clone

public LDAPConnection clone()

Returns a copy of the object with a private context, but sharing the network connection if there is one. The network connection remains

open until all clones have disconnected or gone out of scope. Any connection opened after cloning is private to the object making the connection.

5.4.3. getAuthenticationDN

```
public String getAuthenticationDN()
```

Returns the distinguished name (DN) used for authentication by this object.

5.4.4. getAuthenticationPassword

```
public String getAuthenticationPassword()
```

Returns the password used for simple authentication by this object.

5.4.5. getHost

```
public String getHost()
```

Returns the host name of the LDAP server to which the object is or was last connected, in the format originally specified.

5.4.6. getPort

```
public int getPort()
```

Returns the port number of the LDAP server to which the object is or was last connected.

5.4.7. getProperty

```
public Object getProperty(String name) throws LDAPException
```

Gets a property of a connection object.

Parameters are:

name	Name of the property to be returned. The following read-only properties are available for any given connection:

LDAP_PROPERTY_SDK	The version of this SDK, as a Float data type.
LDAP_PROPERTY_PROTOCOL	The highest supported version of the LDAP protocol, as a Float data type.
LDAP_PROPERTY_SECURITY	A comma-separated list of the types of authentication supported, as a String.

Other properties may be available in particular implementations of the class, and used to modify operations such as search.

An LDAPException is thrown if the requested property is not available.

5.4.8. getSearchConstraints

```
public LDAPSearchConstraints getSearchConstraints()
```

Returns the set of search constraints that apply to all searches performed through this connection (unless a different set of search constraints is specified when calling the search method).

Note that the getOption method can be used to get individual constraints (rather than getting the entire set of constraints).

Typically, the getSearchConstraints method is used to create a slightly different set of search constraints to apply to a particular search.

5.4.9. getSocketFactory

```
public static LDAPSocketFactory getSocketFactory()
```

Returns the default LDAPSocketFactory used to establish a connection to a server.

5.4.10. isAuthenticated

```
public boolean isAthenticated()
```

Indicates whether the object has authenticated to the connected LDAP server.

5.4.11. isConnected

```
public boolean isConnected()
```

Indicates whether the connection represented by this object is open at this time.

5.4.12. read

```
public static LDAPEntry read(LDAPUrl toGet) throws LDAPException
```

Reads the entry specified by the LDAP URL.

When this method is called, a new connection is created automatically, using the host and port specified in the URL. After finding the entry, the method closes the connection (in other words, it disconnects from the LDAP server).

If the URL specifies a filter and scope, these are not used. Of the information specified in the URL, this method only uses the LDAP host name and port number, the base distinguished name (DN), and the list of attributes to return.

The method returns the entry specified by the base DN.

Parameters are:

toGet	LDAP URL specifying the entry to read.

5.4.13. search

```
public static LDAPSearchResults search(LDAPUrl toGet) throws LDAPEx-
ception
```

Performs the search specified by the LDAP URL, returning an enumerable LDAPSearchResults object.

```
public static LDAPSearchResults search(LDAPUrl toGet,
                                       LDAPSearchConstraints cons)
                                       throws LDAPException
```

Perfoms the search specified by the LDAP URL. This method also allows specifying constraints for the search (such as the maximum number of entries to find or the maximum time to wait for search results).

As part of the search constraints, a choice can be made as to whether to have the results delivered all at once or in smaller batches. If the results are to be delivered in smaller batches, each iteration blocks only until the next batch of results is returned.

Parameters are:

toGet	LDAP URL specifying the entry to read.
cons	Constraints specific to the search.

5.4.14. setOption

```
public void setOption(int option,
                      Object value)
                      throws LDAPException
```

Sets the value of the specified option for this LDAPConnection object.

These options represent the default search constraints for the current connection. Some of these options are also propagated through the LDAPSearchConstraints, which can be obtained from the connection object with the getSearchConstraints method.

The option that is set here applies to all subsequent searches performed through the current connection, unless it is overridden with an LDAPSearchConstraints at the time of search.

To set a constraint only for a particular search, create an LDAPSearchConstraints object with the new constraints and pass it to the LDAPConnection.search method.

Parameters are:

option One of the following options:

Option	Type	Description
LDAPv2.DEREF	Integer	Specifies under what circumstances the object dereferences aliases. By default, the value of this option is

LDAPv2.DEREF_NEVER.

Legal values for this option are:

LDAPv2.DEREF_NEVER	Aliases are never dereferenced.
LDAPv2.DEREF_FINDING	aliases are dereferenced when finding the starting point for the search (but not when searching under that starting entry).
LDAPv2.DEREF_SEARCHING	Aliases are dereferenced when searching the entries beneath the starting point of the search (but not when finding the starting entry).
LDAPv2.DEREF_ALWAYS	Aliases are always dereferenced (both when finding the starting point for the search and when searching under that starting entry).

LDAPv2.SIZELIMIT	Integer	Specifies the maximum number of search results to return. If this option is set to 0, there is no maximum limit. By default, the value of this option is 1000.
LDAPv2.TIMELIMIT	Integer	Specifies the maximum number of milliseconds to wait for results before timing out. If this option is set to 0, there is no maximum time limit. The actual granularity of the timeout depends on the implementation.

		By default, the value of this option is 0.
LDAPv2.REFERRALS	Boolean	Specifies whether or not the client follows referrals automatically. If true, the client follows referrals automatically. If false, an LDAPReferralException is raised when a referral is detected. Referrals of any type other to an LDAP server (i.e. a referral URL other than ldap://something) are ignored on automatic referral following. By default, the value of this option is false.
LDAPv2.REFERRALS_REBIND_PROC	LDAPRebind	Specifies an object that implements the LDAPRebind interface. A user of the class library must define this class and the getRebindAuthentication method that will be used to get the distinguished name and password to use for authentication. If this value is null and REFERRALS is true, referrals will be followed with anonymous (= no) authentication. By default, the value of this option is null.
LDAPv2.REFERRALS_HOP_LIMIT	Integer	Specifies the maximum

		number of referrals in a sequence that the client will follow. For example, if REFERRALS_HOP_LIMIT is 5, the client will follow no more than 5 referrals in a row when resolving a single LDAP request. The default value of this option is 10.
LDAPv2.BATCHSIZE	Integer	Specifies the number of search results to return at a time. For example, if BATCHSIZE is 1, enumerating an LDAPSearchResults will block only until one entry is available. If it is 0, enumerating will block until all entries have been retrieved from the server. The default value of this option is 1.

value	The value to assign to the option. The value must be the java.lang object wrapper for the appropriate parameter (e.g. boolean->Boolean, int->Integer) .

5.4.15. setProperty

```
public void setProperty(String name, Object value) throws LDAPException
```

Sets a property of a connection object.

No property names have been defined at this time, but the mechanism is in place in order to support revisional as well as dynamic extensions to operation modifiers.

Parameters are:

name	Name of the property to set.
value	Value to assign to the property.

An LDAPException is thrown if the specified property is not supported.

5.4.16. setSocketFactory

```
public static void setSocketFactory(LDAPSocketFactory factory)
```

Establishes the default LDAPSocketFactory used to establish a connection to a server.

This method is implemented as once-only. It is useful to be able to change the run-time connection behavior of a whole application with a single instruction, but the results would be confusing, and the side-effects dangerous, if the global default factory could be changed at arbitrary times by different threads. It should be called before the first connect(). If called (for the first time) after connecting, the new factory will not be used until/unless a new connection is attempted with the object.

A typical usage would be:

```
    if (usingTLS) {
        LDAPConnection.setSocketFactory(myTLSFactory);
    }
    ...
    LDAPConnection conn = new LDAPConnection();
    conn.connect(myHost, myPort);
```

In this example, connections are constructed with the default LDAPSocketFactory. At application start-up time, the default may be set to use a particular provided TLS socket factory.

Parameters are:

factory	A factory object which can construct socket connections for an LDAPConnection.

5.5. public class LDAPControl implements Cloneable

An LDAPControl encapsulates optional additional parameters or constraints to be applied to LDAP operations. If set as a Server Control,

it is sent to the server along with operation requests. If set as a Client Control, it is not sent to the server, but rather interpreted locally by the client. LDAPControl is an LDAPv3 extension, and is not supported in an LDAPv2 environment.

5.5.1. Constructors

```
   public LDAPControl(String id,
                      boolean critical,
                      byte vals[])
```

Parameters are:

id	The type of the Control, as a string.
critical	True if the LDAP operation should be discarded if the server does not support this Control.
vals	Control-specific data.

5.5.2. getID

```
   public String getID()
```

Returns the identifier of the control.

5.5.3. isCritical

```
   public boolean isCritical()
```

Returns true if the control must be supported for an associated operation to be executed.

5.5.4. getValue

```
   public byte[] getValue()
```

Returns the control-specific data of the object.

5.6. public class LDAPDN

A utility class representing a distinguished name (DN).

5.6.1. explodeDN

```
public static String[] explodeDN(String dn,
                                 boolean noTypes)
```

Returns the individual components of a distinguished name (DN).

Parameters are:

dn	Distinguished name, e.g. "cn=Babs Jensen,ou=Accounting,o=Acme,c=us"
noTypes	If true, returns only the values of the components, and not the names, e.g. "Babs Jensen", "Accounting", "Acme", "us" - instead of "cn=Babs Jensen", "ou=Accounting", "o=Acme", and "c=us".

5.6.2. explodeRDN

```
public static String[] explodeRDN(String rdn,
                                  boolean noTypes)
```

Returns the individual components of a relative distinguished name (RDN).

Parameters are:

rdn	Relative distinguished name, i.e. the left-most com- ponent of a distinguished name.
noTypes	If true, returns only the values of the components, and not the names.

5.7. public class LDAPEntry

An LDAPEntry represents a single entry in a directory, consisting of a distinguished name (DN) and zero or more attributes. An instance of LDAPEntry is created in order to add an entry to a Directory, and instances are returned on a search by enumerating an LDAPSearchResults.

5.7.1. Constructors

```
public LDAPEntry()
```

Constructs an empty entry.

```
public LDAPEntry(String dn)
```

Constructs a new entry with the specified distinguished name and with an empty attribute set.

```
public LDAPEntry(String dn
                 LDAPAttributeSet attrs)
```

Constructs a new entry with the specified distinguished name and set of attributes.

Parameters are:

dn	The distinguished name of the new entry. The value is not validated. An invalid distinguished name will cause adding of the entry to a directory to fail.
attrs	The initial set of attributes assigned to the entry.

5.7.2. getAttribute

```
public LDAPAttribute[] getAttribute(String attrName)
```

Returns the attribute matching the specified attrName. For example, getAttribute("cn") returns only the "cn" attribute; getAttribute("cn;lang-en") returns only the "cn;lang-en" attribute. In both cases, null is returned if there is no exact match to the specified attrName.

```
public LDAPAttribute[] getAttribute(String attrName, String lang)
```

Returns a single best-match attribute, or none if no match is available in the entry.

LDAP version 3 allows adding a subtype specification to an attribute name. "cn;lang-ja", for example, indicates a Japanese language subtype of the "cn" attribute. "cn;lang-ja-JP-kanji" may be a subtype of "cn;lang-ja". This feature may be used to provide multiple localizations in the same Directory. For attributes which do not vary among localizations, only the base attribute may be stored, whereas for others there may be varying degrees of specialization.

getAttribute(attrName,lang) returns the subtype that matches attrName and that best matches lang. If there are subtypes other than "lang" subtypes included in attrName, e.g. "cn;binary", only attributes with all of those subtypes are returned. If lang is null or empty, the

method behaves as getAttribute(attrName). If there are no matching attributes, null is returned.

Example:

```
   Assume the entry contains only the following attributes:
      cn;lang-en
      cn;lang-ja-JP-kanji
      sn
   getAttribute( "cn" ) returns null.
   getAttribute( "sn" ) returns the "sn" attribute.
   getAttribute( "cn", "lang-en-us" ) returns the "cn;lang-en" attribute.
   getAttribute( "cn", "lang-en" ) returns the "cn;lang-en" attribute.
   getAttribute( "cn", "lang-ja" ) returns null.
   getAttribute( "sn", "lang-en" ) returns the "sn" attribute.
```

Parameters are:

attrName	The name of an attribute to retrieve, with or without subtype specification(s). "cn", "cn;phonetic", and "cn;binary" are valid attribute names.
lang	A language specification as in [14], with optional subtypes appended using "-" as separator. "lang-en", "lang-en-us", "lang-ja", and "lang-ja-JP-kanji" are valid language specification.

5.7.3. getAttributeSet

```
public LDAPAttributeSet getAttributeSet()
```

Returns the attribute set of the entry. All base and subtype variants of all attributes are returned. The LDAPAttributeSet returned may be empty if there are no attributes in the entry.

```
public LDAPAttributeSet getAttributeSet(String subtype)
```

Returns an attribute set from the entry, consisting of only those attributes matching the specified subtype(s). This may be used to extract only a particular language variant subtype of each attribute, if it exists. "subtype" may be, for example, "lang-ja", "binary", or "lang-ja;phonetic". If more than one subtype is specified, separated with a semicolon, only those attributes with all of the named subtypes will be returned. The LDAPAttributeSet returned may be empty if there are no matching attributes in the entry.

Parameters are:

subtype	One or more subtype specification(s), separated with semicolons. "lang-ja" and "lang-en;phonetic" are valid subtype specifications.

5.7.4. getDN

```
public String getDN()
```

Returns the distinguished name of the entry.

5.8. public class LDAPExtendedOperation

An LDAPExtendedOperation encapsulates an ID which uniquely identifies a particular extended operation, known to a particular server, and the data associated with the operation.

5.8.1. Constructors

```
public LDAPExtendedOperation(String oid,
                             byte[] vals)
```

Constructs a new object with the specified object ID and data.

Parameters are:

oid	The unique identifier of the operation.
vals	The operation-specific data of the operation/

5.8.2. getID

```
public String getID()
```

Returns the unique identifier of the operation.

5.8.3. getValue

```
public byte[] getValue()
```

Returns the operation-specific data (not a copy, a reference).

5.9. public interface LDAPEntryComparator

An object of this class can implement arbitrary sorting algorithms for search results.

5.9.1. isGreater

```
public boolean isGreater(LDAPEntry entry1, LDAPEntry entry2)
```

Returns true if entry1 is to be considered greater than or equal to entry2, for the purpose of sorting.

Parameters are:

entry1	Target entry for comparison.
entry2	Entry to be compared to.

5.10. public class LDAPException extends Exception

Thrown to indicate that an error has occurred. 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).

Most errors that occur throw this type of exception. The getLDAPResultCode() method returns the specific result code, which can be compared against standard LDAP result codes as defined in [11], section 4.

5.10.1. Constructors

```
public LDAPException()
```

Constructs a default exception with no specific error information.

```
public LDAPException(String message,
                     int resultCode)
```

Constructs an exception with an error code and a specified string as additional information.

Parameters are:

message	The additional error information.
resultCode	The result code returned

5.10.2. getLDAPErrorMessage

```
public String getLDAPErrorMessage()
```

Returns the error message, if this message is available (that is, if this message was set). If the message was not set, this method returns null.

5.10.3. getLDAPResultCode

```
public int getLDAPResultCode()
```

Returns the result code from the exception. The codes are defined as public final static int members of this class. If the exception is a result of error information returned from a directory operation, the code will be one of those defined in [11]. Otherwise, a local error code is returned (see "Error codes" below).

5.10.4. getMatchedDN

```
public String getMatchedDN()
```

Returns the part of a submitted distinguished name which could be matched by the server. If the exception was caused by a local error, such as no server available, the return value is null. If the exception resulted from an operation being executed on a server, the value is an empty String except when the result of the operation was one of the following:

```
NO_SUCH_OBJECT
ALIAS_PROBLEM
INVALID_DN_SYNTAX
ALIAS_DEREFERENCING_PROBLEM
```

5.10.5. Error codes

See [11] and [7] for a discussion of the meanings of the codes.

```
ADMIN_LIMIT_EXCEEDED
AFFECTS_MULTIPLE_DSAS
ALIAS_DEREFERENCING_PROBLEM
```

```
     ALIAS_PROBLEM
     ATTRIBUTE_OR_VALUE_EXISTS
     AUTH_METHOD_NOT_SUPPORTED
     BUSY
     COMPARE_FALSE
     COMPARE_TRUE
     CONFIDENTIALITY_REQUIRED
     CONSTRAINT_VIOLATION
     ENTRY_ALREADY_EXISTS
     INAPPROPRIATE_AUTHENTICATION
     INAPPROPRIATE_MATCHING
     INSUFFICIENT_ACCESS_RIGHTS
     INVALID_ATTRIBUTE_SYNTAX
     INVALID_CREDENTIALS
     INVALID_DN_SYNTAX
     IS_LEAF
     LDAP_PARTIAL_RESULTS
     LOOP_DETECT
     NAMING_VIOLATION
     NO_SUCH_ATTRIBUTE
     NO_SUCH_OBJECT
     NOT_ALLOWED_ON_NONLEAF
     NOT_ALLOWED_ON_RDN
     OBJECT_CLASS_MODS_PROHIBITED
     OBJECT_CLASS_VIOLATION
     OPERATIONS_ERROR
     OTHER
     PROTOCOL_ERROR
     REFERRAL
     SIZE_LIMIT_EXCEEDED
     STRONG_AUTH_NOT_SUPPORTED
     STRONG_AUTH_REQUIRED
     SUCCESS
     TIME_LIMIT_EXCEEDED
     UNAVAILABLE
     UNAVAILABLE_CRITICAL_EXTENSION
     UNDEFINED_ATTRIBUTE_TYPE
     UNWILLING_TO_PERFORM
```

Local errors, resulting from actions other than an operation on a server, are among the following:

```
     CONNECT_ERROR
     PARAM_ERROR
     SERVER_DOWN
```

5.11. public class LDAPModification

A single change specification for an LDAPAttribute.

5.11.1. Constructors

```
public LDAPModification(int op,
                        LDAPAttribute attr)
```

Specifies a modification to be made to an attribute.

Parameters are:

op	The type of modification to make, which can be one of the following:

LDAPModification.ADD	The value should be added to the attribute
LDAPModification.DELETE	The value should be removed from the attribute
LDAPModification.REPLACE	The value should replace all existing values of the attribute

attr	The attribute (possibly with values) to be modified.

5.11.2. getAttribute

```
public LDAPAttribute getAttribute()
```

Returns the attribute (possibly with values) to be modified.

5.11.3. getOp

```
public int getOp()
```

Returns the type of modification specified by this object.

5.12. public class LDAPModificationSet

A collection of modifications to be made to the attributes of a single entry.

5.12.1. Constructors

```
public LDAPModificationSet()
```

Constructs a new, empty set of modifications.

5.12.2. add

```
public synchronized void add(int op,
                             LDAPAttribute attr)
```

Specifies another modification to be added to the set of modifications.

Parameters are:

op	The type of modification to make, as described for LDAPModification.
attr	The attribute (possibly with values) to be modified.

5.12.3. elementAt

```
public LDAPModification elementAt(int index)
                            throws ArrayIndexOutOfBoundsException
```

Retrieves a particular LDAPModification object at the position specified by the index.

Parameters are:

index	Index of the modification to get.

5.12.4. remove

```
public synchronized void remove(String name)
```

Removes the first attribute with the specified name in the set of modifications.

Parameters are:

name	Name of the attribute to be removed.

5.12.5. removeElementAt

```
   public void removeElementAt(int index)
                            throws ArrayIndexOutOfBoundsException
```

Removes a particular LDAPModification object at the position specified by the index.

index	Index of the modification to remove.

5.12.6. size

```
   public int size()
```

Retrieves the number of LDAPModification objects in this set.

5.13. public class LDAPRebindAuth

Represents information used to authenticate the client in cases where the client follows referrals automatically.

5.13.1. Constructors

```
   public LDAPRebindAuth(String dn,
                         String password)
```

Constructs information that is used by the client for authentication when following referrals automatically.

5.13.2. getDN

```
   public String getDN()
```

Returns the distinguished name to be used for reauthentication on automatic referral following.

5.13.3. getPassword

```
   public String getPassword()
```

Returns the password to be used for reauthentication on automatic referral following.

5.14. public class LDAPReferralException extends LDAPException

This exception, derived from LDAPException, is thrown when a server returns a referral and automatic referral following has not been enabled.

5.14.1. Constructors

```
   public LDAPReferralException()
```

Constructs a default exception with no specific error information.

```
   public LDAPReferralException(String message)
```

Constructs a default exception with a specified string as additional information. This form is used for lower-level errors.

```
   public LDAPReferralException(String message,
                                int resultCode,
                                String serverMessage)
```

Parameters are:

message	The additional error information.
resultCode	The result code returned
serverMessage	Error message specifying additional information from the server.

5.14.2. getURLs

```
   public LDAPUrl[] getURLs()
```

Gets the list of referrals (LDAP URLs to other servers) returned by the LDAP server. This exception is only thrown, and therefor the URL list only available, if automatic referral following is not enabled. The referrals may include URLs of a type other than ones for an LDAP server (i.e. a referral URL other than ldap://something).

5.15. public class LDAPSearchConstraints

A set of options to control a search operation. There is always an

LDAPSearchConstraints associated with an LDAPConnection object; its values can be changed with LDAPConnection.setOption, or overridden by passing an LDAPSearchConstraints object to the search operation.

5.15.1. Constructors

```
   public LDAPSearchConstraints()
```

Constructs an LDAPSearchConstraints object that specifies the default set of search constraints.

```
   public LDAPSearchConstraints(int msLimit,
                                int dereference,
                                int maxResults,
                                boolean doReferrals,
                                int batchSize,
                                LDAPRebind rebind_proc,
                                int hop_limit)
```

Constructs a new LDAPSearchConstraints object and allows specifying the search constraints in that object.

Parameters are:

msLimit	Maximum time in milliseconds to wait for results (0 by default, which means that there is no maximum time limit).
dereference	Specifies when aliases should be dereferenced. Must be either LDAP_DEREF_NEVER, LDAP_DEREF_FINDING, LDAP_DEREF_SEARCHING, or LDAP_DEREF_ALWAYS from LDAPv2 (LDAPv2.LDAP_DEREF_NEVER by default).
maxResults	Maximum number of search results to return (1000 by default).
doReferrals	Specify true to follow referrals automatically, or false to throw an LDAPReferralException error if the server sends back a referral (false by default)
batchSize	Specify the number of results to block on during enumeration. 0 means to block until all results are in (1 by default).
rebind_proc	Specifies an object of the class that implements the LDAPRebind interface. The object will be used when

	the client follows referrals automatically. The object provides a method for getting the distinguished name and password used to authenticate to another LDAP server during a referral. This field is null by default.
hop_limit	Maximum number of referrals to follow in a sequence when attempting to resolve a request, when doing automatic referral following.

5.15.2. getBatchSize

```
public int getBatchSize()
```

Returns the number of results to block on during enumeration of search results. This should be 0 if intermediate results are not needed, and 1 if results are to be processed as they come in.

5.15.3. getDereference

```
public int getDereference()
```

Specifies when aliases should be dereferenced. Returns either LDAP_DEREF_NEVER, LDAP_DEREF_FINDING, LDAP_DEREF_SEARCHING, or LDAP_DEREF_ALWAYS from LDAPv2.

5.15.4. getHopLimit

```
public int getHopLimit()
```

Returns the maximum number of hops to follow during automatic referral following.

5.15.5. getMaxResults

```
public int getMaxResults()
```

Returns the maximum number of search results to be returned; 0 means no limit.

5.15.6. getRebindProc

```
public LDAPRebind getRebindProc()
```

Returns the object that provides the method for getting authentication information.

5.15.7. getReferrals

```
public boolean getReferrals()
```

Specifies whether nor not referrals are followed automatically. Returns true if referrals are to be followed automatically, or false if referrals throw an LDAPReferralException.

5.15.8. getTimeLimit

```
public int getTimeLimit()
```

Returns the maximum number of milliseconds to wait for any operation under these search constraints. If 0, there is no maximum time limit on waiting for the operation results. The actual granularity of the timeout depends on the implementation.

5.15.9. setBatchSize

```
public void setBatchSize(int batchSize)
```

Sets the suggested number of results to block on during enumeration of search results. This should be 0 if intermediate results are not needed, and 1 if results are to be processed as they come in. The default is 1.

Parameters are:

batchSize	Blocking size on search enumerations.

5.15.10. setDereference

```
public void setDereference(int dereference)
```

Sets a preference indicating whether or not aliases should be dereferenced, and if so, when.

Parameters are:

dereference	Either LDAP_DEREF_NEVER, LDAP_DEREF_FINDING, LDAP_DEREF_SEARCHING, or LDAP_DEREF_ALWAYS from

LDAPv2.

5.15.11. setHopLimit

public void setHopLimit(int hop_limit)

Sets the maximum number of hops to follow in sequence during automatic referral following. The default is 5.

Parameters are:

hop_limit	Maximum number of chained referrals to follow automatically.

5.15.12. setMaxResults

public void setMaxResults(int maxResults)

Sets the maximum number of search results to be returned; 0 means no limit. The default is 1000.

Parameters are:

maxResults	Maxumum number of search results to return.

5.15.13. setRebindProc

public void setRebindProc(LDAPRebind rebind_proc)

Specifies the object that provides the method for getting authentication information. The default is null. If referrals is set to true, and the rebindProc is null, referrals will be followed with anonymous (= no) authentication.

Parameters are:

rebind_proc	An object that implements LDAPRebind.

5.15.14. setReferrals

public void setReferrals(boolean doReferrals)

Specifies whether nor not referrals are followed automatically, or if referrals throw an LDAPReferralException. Referrals of any type

other to an LDAP server (i.e. a referral URL other than ldap://something) are ignored on automatic referral following. The default is false.

Parameters are:

doReferrals	True to follow referrals automatically.

5.15.15. setTimeLimit

```
public void setTimeLimit(int msLimit)
```

Sets the maximum number of milliseconds to wait for any operation under these search constraints. If 0, there is no maximum time limit on waiting for the operation results. The actual granularity of the timeout depends on the implementation.

Parameters are:

msLimit	Maximum milliseconds to wait.

5.16. public class LDAPSearchResults

An LDAPSearchResults object is returned from a search operation. It implements Enumeration, thereby providing access to all entries retrieved during the operation.

5.16.1. hasMoreElements

```
public boolean hasMoreElements()
```

Specifies whether or not there are more search results in the enumeration. If true, there are more search results.

5.16.2. next

```
public LDAPEntry next() throws LDAPException
```

Returns the next result in the enumeration as an LDAPEntry. If automatic referral following is disabled, and there are one or more referrals among the search results, next() will throw an LDAPReferralException the last time it is called, after all other results have been returned.

5.16.3. nextElement

```
public Object nextElement()
```

Returns the next result in the enumeration as an Object. This the default implementation of Enumeration.nextElement(). The returned value may be an LDAPEntry or an LDAPReferralException.

5.16.4. sort

```
public void sort(LDAPEntryComparator comp)
```

Sorts all entries in the results using the provided comparison object. If the object has been partially or completely enumerated, only remaining elements are sorted. Sorting the results requires that they all be present. This implies that LDAPSearchResults.nextElement() will always block until all results have been retrieved, after a sort operation.

The LDAPCompareAttrNames class is provided to support the common need to collate by a single or multiple attribute values, in ascending or descending order. Examples are:

```
res.sort(new LDAPCompareAttrNames("cn"));

res.sort(new LDAPCompareAttrNames("cn", false));

String[] attrNames = { "sn", "givenname" };
res.sort(new LDAPCompareAttrNames(attrNames));
```

Parameters are:

comp	An object that implements the LDAPEntryComparator interface to compare two objects of type LDAPEntry.

5.17. public interface LDAPSocketFactory

Used to construct a socket connection for use in an LDAPConnection. An implementation of this interface may, for example, provide a TLSSocket connected to a secure server.

5.17.1. makeSocket

```
public Socket makeSocket(String host, int port)
                         throws IOException, UnknownHostException
```

Returns a socket connected using the provided host name and port number.

There may be additional makeSocket methods defined when interfaces to establish TLS and SASL authentication in the java environment have been standardized.

Parameters are:

host	Contains a hostname or dotted string representing the IP address of a host running an LDAP server to connect to.
port	Contains the TCP or UDP port number to connect to or contact. The default LDAP port is 389.

5.18. **public class LDAPSortKey**

Encapsulates parameters for sorting search results.

5.18.1. **Constructors**

```
public LDAPSortKey( String keyDescription )
```

Constructs a new LDAPSortKey object using a, possibly complex, sorting specification.

```
public LDAPSortKey( String key, boolean reverse)
```

Constructs a new LDAPSortKey object using an attribute name and a sort order.

```
public LDAPSortKey( String key, boolean reverse, String matchRule)
```

Constructs a new LDAPSortKey object using an attribute name, a sort order, and a matching rule.

Parameters are:

```
keyDescription  A single attribute specification to sort by. If pre-
                fixed with "-", reverse order sorting is requested. A
                matching rule OID may be appended following ":".
                Examples:
                   "cn"
```

```
                      "-cn"
                      "-cn:1.2.3.4.5"
```

key	An attribute name, e.g. "cn".
reverse	True to sort in reverse collation order.
matchRule	The object ID (OID) of a matching rule used for collation. If the object will be used to request server-side sorting of search results, it should be the OID of a matching rule known to be supported by that server.

5.18.2. getKey

```
   public String getKey()
```

Returns the attribute to be used for collation.

5.18.3. getReverse

```
   public boolean getReverse()
```

Returns true if the sort key specifies reverse-order sorting.

5.18.4. getMatchRule

```
   public String getMatchRule()
```

Returns the OID to be used as matching rule, or null if none is to be used.

5.19. public class LDAPUrl

Encapsulates parameters of an LDAP Url query, as defined in [8]. An LDAPUrl object can be passed to LDAPConnection.search to retrieve search results.

5.19.1. Constructors

```
   public LDAPUrl(String url) throws MalformedURLException
```

Constructs a URL object with the specified string as URL.

```
public LDAPUrl(String host,
               int port,
               String dn)
```

Constructs with the specified host, port, and DN. This form is used to create URL references to a particular object in the directory.

```
public LDAPUrl(String host,
               int port,
               String dn,
               String attrNames[],
               int scope,
               String filter)
```

Constructs a full-blown LDAP URL to specify an LDAP search operation.

Parameters are:

url	An explicit URL string, e.g. "ldap://ldap.acme.com:80/o=Ace%20Industry,c=us?cn,sn?sub?(objectclass=inetOrgPerson)".
host	Host name of LDAP server, or null for "nearest X.500/LDAP".
port	Port number for LDAP server (use LDAPConnection.DEFAULT_PORT for default port).
dn	Distinguished name of object to fetch.
attrNames	Names of attributes to retrieve. null for all attri- butes.
scope	Depth of search (in DN namespace). Use one of SCOPE_BASE, SCOPE_ONE, SCOPE_SUB from LDAPv2.

5.19.2. **decode**

```
public static String decode(String URLEncoded) throws MalformedURLEx-
ception
```

Decodes a URL-encoded string. Any occurences of %HH are decoded to the hex value represented. However, this routine does NOT decode "+"

into " ". See [10] for details on URL encoding/decoding.

Parameters are:

URLEncoded	String to decode.

5.19.3. encode

```
public static String encode(String toEncode)
```

Encodes an arbitrary string. Any illegal characters are encoded as %HH. However, this routine does NOT encode " " into "+".

Parameters are:

toEncode	String to encode.

5.19.4. getAttributes

```
public String[] getAttributeArray()
```

Return an array of attribute names specified in the URL

5.19.5. getAttributes

```
public Enumeration getAttributes()
```

Return an Enumerator for the attribute names specified in the URL

5.19.6. getDN

```
public String getDN()
```

Return the distinguished name encapsulated in the URL.

5.19.7. getFilter

```
public String getFilter()
```

Returns the search filter [8], or the default filter - (objectclass=*) - if none was specified.

5.19.8. getHost

```
public String getHost()
```

Returns the host name of the LDAP server to connect to.

5.19.9. getPort

```
public int getPort()
```

Returns the port number of the LDAP server to connect to.

5.19.10. getUrl

```
public String getUrl()
```

Returns a valid string representation of this LDAP URL.

5.20. public interface LDAPv2

As a mechanism to support planned and future LDAP protocol extensions, functionality is defined in an interface - LDAPv2, corresponding to version 2 of the LDAP protocol. LDAPConnection must implement at least LDAPv2, and may implement LDAPv3. Applications can test for support of these protocol levels in a given package with the instanceof operator.

5.20.1. add

```
public void add(LDAPEntry entry) throws LDAPException
```

Adds an entry to the directory.

Parameters are:

entry	LDAPEntry object specifying the distinguished name and attributes of the new entry.

5.20.2. authenticate

```
public void authenticate(String dn,
                         String passwd)
                         throws LDAPException
```

Authenticates to the LDAP server (that the object is currently connected to) using the specified name and password. If the object has

been disconnected from an LDAP server, this method attempts to reconnect to the server. If the object had already authenticated, the old authentication is discarded.

Parameters are:

dn	If non-null and non-empty, specifies that the connection and all operations through it should be authenticated with dn as the distinguished name.
passwd	If non-null and non-empty, specifies that the connection and all operations through it should be authenticated with dn as the distinguished name and passwd as password.

5.20.3. compare

```
public boolean compare(String dn,
                       LDAPAttribute attr)
                       throws LDAPException
```

Checks to see if an entry contains an attribute with a specified value. Returns true if the entry has the value, and false if the entry does not have the value or the attribute.

Parameters are:

dn	The distinguished name of the entry to use in the comparison.
attr	The attribute to compare against the entry. The method checks to see if the entry has an attribute with the same name and value as this attribute.

5.20.4. connect

```
public void connect(String host,
                    int port)
                    throws LDAPException
```

Connects to the specified host and port. If this LDAPConnection object represents an open connection, the connection is closed first before the new connection is opened. At this point there is no authentication, and any operations will be conducted as an anonymous client.

```
public void connect(String host,
                    int port,
                    String dn,
                    String passwd)
                    throws LDAPException
```

Connects to the specified host and port and uses the specified DN and password to authenticate to the server. If this LDAPConnection object represents an open connection, the connection is closed first before the new connection is opened. This is equivalent to connect(host, port) followed by authenticate(dn, passwd).

Parameters are:

host	Contains a hostname or dotted string representing the IP address of a host running an LDAP server to connect to. Alternatively, it may contain a list of host names, space-delimited. Each host name may include a trailing colon and port number. In the case where more than one host name is specified, each host name in turn will be contacted until a connection can be established. Examples:

```
"directory.knowledge.com"
"199.254.1.2"
"directory.knowledge.com:1050 people.catalog.com 199.254.1.2"
```

port	Contains the TCP or UDP port number to connect to or contact. The default LDAP port is 389. "port" is ignored for any host name which includes a colon and port number.
dn	If non-null and non-empty, specifies that the connection and all operations through it should be authenticated with dn as the distinguished name.
passwd	If non-null and non-empty, specifies that the connection and all operations through it should be authenticated with dn as the distinguished name and passwd as password.

5.20.5. delete

```
public void delete(String dn) throws LDAPException
```

Deletes the entry for the specified DN from the directory.

Parameters are:

dn	Distinguished name of the entry to modify.

5.20.6. disconnect

```
public synchronized void disconnect() throws LDAPException
```

Disconnects from the LDAP server. Before the object can perform LDAP operations again, it must reconnect to the server by calling connect.

5.20.7. getOption

```
public Object getOption(int option) throws LDAPException
```

Returns the value of the specified option for this object.

Parameters are:

option	See LDAPConnection.setOption for a description of valid options.

5.20.8. modify

```
public void modify(String dn,
                   LDAPModification mod)
                   throws LDAPException
```

Makes a single change to an existing entry in the directory (for example, changes the value of an attribute, adds a new attribute value, or removes an existing attribute value).

The LDAPModification object specifies both the change to be made and the LDAPAttribute value to be changed.

```
public void modify(String dn,
                   LDAPModificationSet mods)
                   throws LDAPException
```

Makes a set of changes to an existing entry in the directory (for example, changes attribute values, adds new attribute values, or removes existing attribute values).

Parameters are:

dn	Distinguished name of the entry to modify.
mod	A single change to be made to the entry.
mods	A set of changes to be made to the entry.

5.20.9. read

```
public LDAPEntry read(String dn) throws LDAPException
```

Reads the entry for the specified distiguished name (DN) and retrieves all attributes for the entry.

```
public LDAPEntry read(String dn,
                      String attrs[])
                      throws LDAPException
```

Reads the entry for the specified distinguished name (DN) and retrieves only the specified attributes from the entry.

Parameters are:

dn	Distinguished name of the entry to retrieve.
attrs	Names of attributes to retrieve.

5.20.10. rename

```
public void rename(String dn,
                   String newRdn,
                   boolean deleteOldRdn)
                   throws LDAPException
```

Renames an existing entry in the directory.

Parameters are:

dn	Current distinguished name of the entry.
newRdn	New relative distinguished name for the entry.
deleteOldRdn	If true, the old name is not retained as an attribute value.

5.20.11. search

```
public LDAPSearchResults search(String base,
                                int scope,
                                String filter,
                                String attrs[],
                                boolean attrsOnly)
                                throws LDAPException
```

Performs the search specified by the parameters.

```
public LDAPSearchResults search(String base,
                                int scope,
                                String filter,
                                String attrs[],
                                boolean attrsOnly,
                                LDAPSearchConstraints cons)
                                throws LDAPException
```

Performs the search specified by the parameters, also allowing specification of constraints for the search (such as the maximum number of entries to find or the maximum time to wait for search results).

As part of the search constraints, the function allows specifying whether or not the results are to be delivered all at once or in smaller batches. If specified that the results are to be delivered in smaller batches, each iteration blocks only until the next batch of results is returned.

Parameters are:

base	The base distinguished name to search from.
scope	The scope of the entries to search. The following are the valid options:

LDAPv2.SCOPE_BASE	Search only the base DN
LDAPv2.SCOPE_ONE	Search only entries under the base DN
LDAPv2.SCOPE_SUB	Search the base DN and all entries within its subtree

filter	Search filter specifying the search criteria, as

	defined in [3].
attrs	Names of attributes to retrieve.
attrsOnly	If true, returns the names but not the values of the attributes found. If false, returns the names and values for attributes found
cons	Constraints specific to the search.

5.20.12. setOption

```
   public void setOption(int option,
                         Object value)
                         throws LDAPException
```

Sets the value of the specified option for this LDAPConnection object.

See LDAPConnection.setOption for an implementation.

5.21. public interface LDAPv3 extends LDAPv2

LDAPv3 extends LDAPv2 by adding support for features of version 3 of the LDAP protocol. LDAPConnection implements at least LDAPv2, and may also implement LDAPv3. Applications can test for support of these protocol levels in a given package with the instanceof operator.

5.21.1. Preferred Language

A preferred language, specified as in [14], can be set using setOption. A Preferred Language Server Control is constructed and sent to the server with all operations. If the server supports the control, results returned on search() or read() will be filtered using the control, as per [15], e.g.

```
      ld.setOption( LDAPv3.PREFERRED_LANGUAGE, "lang-en" );
```

5.21.2. authenticate

```
   public void authenticate(int version,
                            String dn,
                            String passwd)
                            throws LDAPException
```

Authenticates to the LDAP server (that the object is currently connected to) using the specified name and password, with the specified LDAP protocol version. If the server does not support the requested protocol version, an exception is thrown. If the object has been disconnected from an LDAP server, this method attempts to reconnect to the server. If the object had already authenticated, the old authentication is discarded.

Parameters are:

version	LDAP protocol version requested: currently 2 or 3.
dn	If non-null and non-empty, specifies that the connection and all operations through it should be authenticated with dn as the distinguished name.
passwd	If non-null and non-empty, specifies that the connection and all operations through it should be authenticated with dn as the distinguished name and passwd as password.

5.21.3. authenticate

```
public void authenticate(String dn,
                         byte[] credentials,
                         String[] mechanisms,
                         Properties props,
                         SaslAuthenticationCallback authCb)
                         throws LDAPException
```

Authenticates to the LDAP server using SASL authentication mechanisms. Mechanisms will be tried in the order provided. If a mechanism requires additional information (e.g. credentials) to procede, and authCb is not null, authCb's request() method will be called. An LDAPException is thrown if authentication fails for all specified mechanisms.

Parameters are:

dn	Distinguished name to authenticate as
credentials	Initial credentials to use on first authentication request. The contents are specific to a SASL mechanism.
props	Properties to be used for authentication. Some are only meaningful to a subset of all SASL mechanisms.

The following is a partial list of properties and sample values:

security.userid	"default"
security.policy.encryption	"true"
security.policy.sign_without_encryption	"false"
security.policy.no_encryption	"false"
security.policy.encryption.minimum	"40"
security.policy.encryption.maximum	"128"
security.policy.server_authentication	"true"
security.server.fqdn	"safe.mcom.com"
security.maxbuffer	"4096"
security.ip.local	"192.68.1.10"
security.ip.remote	"192.68.1.50"

authCb An object implementing the SaslAuthenticationCallback interface, capable of returning additional information to a SASL mechanism driver if necessary. This may or may not involve interactively prompting the user for this information. The parameter may be null, indicating that the application will not provide additional information.

5.21.4. extendedOperation

```
public LDAPExtendedOperation extendedOperation(
                                LDAPExtendedOperation op )
                                throws LDAPException
```

Provides a means to access extended, non-mandatory operations offered by a particular LDAP version 3 compliant server.

Returns an operation-specific object, containing an ID and an Octet String or BER-encoded value(s).

Parameters are:

op	Object which contains an identifier of the extended operation, which should be one recognized by the particular server this client is connected to, and an operation-specific sequence of Octet String or BER-encoded value(s).

5.21.5. getResponseControls

```
public LDAPControl[] getResponseControls()
```

Returns the latest Server Controls returned by a Directory Server with a response to an LDAP request from the current thread, or null if the latest response contained no Server Controls.

5.21.6. rename

```
public void rename(String dn,
                   String newRdn,
                   String newParentdn,
                   boolean deleteOldRdn)
                   throws LDAPException
```

Renames an existing entry in the directory, possibly repositioning it in the directory tree.

Parameters are:

dn	Current distinguished name of the entry.
newRdn	New relative distinguished name for the entry.
newParentdn	Distinguished name of the existing entry which is to be the new parent of the entry.
deleteOldRdn	If true, the old name is not retained as an attribute value.

5.22. public interface SaslAuthenticationCallback

Note: this interface is not part of the LDAP classes. It is presented here to clarify use of SASL mechanisms in authentication using the LDAP classes.

An application may implement this interface to allow a SASL mechanism driver to obtain additional information (e.g. credentials) as needed

during authentication. The implementation may or may not include interactively prompting a user.

5.22.1. request

```
   public byte[] request(String prompt,
                         String type)
                         throws SaslException
```

Parameters are:

prompt	A prompt that may be presented to a user as a guide to entering the requested information, or may be used as a key to forming the user interface for requesting the information, or may be ignored.
type	An identifier of type of information requested by the SASL mechanism driver, e.g. "password".

5.23. Client and Server Controls

LDAPv3 operations can be extended through the use of controls. Controls may be sent to a server or returned to the client with any LDAP message. These controls are referred to as server controls. The LDAP API also supports a client-side extension mechanism through the use of client controls (these controls affect the behavior of the LDAP API only and are never sent to a server). A common class is used to represent both types of controls - LDAPControl.

Controls are set and retrieved in LDAPConnection with the setOption and getOption methods, using the keys LDAPv3.SERVERCONTROLS and LDAPv3.CLIENTCONTROLS. Either a single LDAPControl or an array may be passed, e.g.

```
      LDAPControl control = new LDAPControl( type, critical, vals );
      ld.setOption( LDAPv3.SERVERCONTROLS, control );
   or
      LDAPControl[] controls = new LDAPControl[2];
      controls[0] = new LDAPControl( type0, critical0, vals0 );
      controls[1] = new LDAPControl( type1, critical1, vals1 );
      ld.setOption( LDAPv3.SERVERCONTROLS, controls );
```

Server controls returned to a client as part of the response to an operation can be obtained with LDAPv3.getResponseControls().

Support for specific controls is defined in a package "controls" subor- dinate to the main LDAP package.

6. Security Considerations

LDAP supports security through protocol-level authentication, using clear-text passwords or other more secure mechanisms. It also supports running over TLS, which provides strong security at the transport layer. This draft does not cover TLS implementations, although it identifies a mechanism for supplying one, through the LDAPSocketFactory interface. An interface is defined for using protocol-independent SASL mechanism drivers for authentication.

7. Acknowledgements

The proposed API builds on earlier work done in collaboration with Thomas Kwan and Stephan Gudmundson, then of of NCware Technologies Corp.

8. Bibliography

[1] The Directory: Selected Attribute Syntaxes. CCITT, Recommendation X.520.

[2] M. Wahl, A. Coulbeck, T. Howes, S. Kille, "Lightweight Directory Access Protocol: Standard and Pilot Attribute Definitions", Internet Draft draft-ietf-asid-ldapv3-attributes-03.txt, October 1996

[3] T. Howes, "A String Representation of LDAP Search Filters," RFC 1960, June 1996.

[4] S. Kille, "A String Representation of Distinguished Names," RFC 1779, March 1995.

[5] S. Kille, "Using the OSI Directory to Achieve User Friendly Naming," RFC 1781, March 1995.

[7] M. Wahl, T. Howes, S. Kille, "Lightweight Directory Access Protocol (v3)", Internet Draft draft-ietf-asid-ldapv3-protocol-04.txt, March 1997.

[8] T. Howes, M. Smith, "An LDAP URL Format", RFC 1959, June 1996.

[9] T. Howes, M. Smith, "The LDAP Application Program Interface", RFC 1823, August 1995.

[10] T. Berners-Lee, L. Masinter, M. McCahill, "Uniform Resource Locators (URL)", RFC 1738, December 1994.

[11] W. Yeong, T. Howes, S. Kille, "Lightweight Directory Access Protocol", RFC 1777, March 1995.

[12] R. Weltman, "The Java LDAP Application Program Interface", Internet Draft draft-weltman-java-ldap-01.txt, April 1997.

[13] R. Weltman, T. Howes, M. Smith, "The Java LDAP Application Program Interface", Internet Draft draft-ietf-asid-ldap-java-api-00.txt, July 1997.

[14] H. Alvestrans, "Tags for the Identification of Languages", Request for Comments 1766, March 1995.

[15] M. Wahl, T. Howes, "Use of Language Codes in LDAPv3", Internet Draft draft-ietf-asid-ldapv3-lang-02.txt, June 1997.

9. Authors' Addresses

Rob Weltman
Netscape Communications Corp.
501 E. Middlefield Rd.
Mountain View, CA 94043
USA
+1 650 937-3301
rweltman@netscape.com

Tim Howes
Netscape Communications Corp.
501 E. Middlefield Rd.
Mountain View, CA 94043
USA
+1 650 937-3419
howes@netscape.com

Mark Smith
Netscape Communications Corp.
501 E. Middlefield Rd.
Mountain View, CA 94043
USA
+1 650 937-3477
mcs@netscape.com

10. Appendix A - Sample java LDAP programs

```
   import netscape.ldap.*;
   import java.util.*;

   public class SearchJensen {
       public static void main( String[] args )
       {
           try {
               LDAPConnection ld = new LDAPConnection();
               /* Connect to server */
               String MY_HOST = "localhost";
               int MY_PORT = 389;
               ld.connect( MY_HOST, MY_PORT );

               /* authenticate to the directory as nobody */
               /* This is not really necessary if explicit authentication
                  is not desired, because there is already anonymous
                  authentication at connect time */
               ld.authenticate( "", "" );

               /* search for all entries with surname of Jensen */
               String MY_FILTER = "sn=Jensen";
               String MY_SEARCHBASE = "o=Ace Industry, c=US";

               LDAPSearchConstraints cons = ld.getSearchConstraints();
               /* Setting the batchSize to one will cause the result
                  enumeration below to block on one result at a time,
                  allowing us to update a list or do other things as
                  results come in. */
               /* We could set it to 0 if we just wanted to get all
                  results and were willing to block until then */
               cons.setBatchSize( 1 );
               LDAPSearchResults res = ld.search( MY_SEARCHBASE,
                                                  LDAPConnection.SCOPE_ONE,
                                                  MY_FILTER,
                                                  null,
                                                  false,
                                                  cons );

               /* Loop on results until finished */
               while ( res.hasMoreElements() ) {

                   /* Next directory entry */
                   LDAPEntry findEntry = (LDAPEntry)res.nextElement();
                   System.out.println( findEntry.getDN() );

                   /* Get the attributes of the entry */
```

```
                   LDAPAttributeSet findAttrs = findEntry.getAttributeSet();
                   Enumeration enumAttrs = findAttrs.getAttributes();
                   System.out.println( "Attributes: " );
                   /* Loop on attributes */
                   while ( enumAttrs.hasMoreElements() ) {
                       LDAPAttribute anAttr =
                           (LDAPAttribute)enumAttrs.nextElement();
                       String attrName = anAttr.getName();
                       System.out.println( "" + attrName );
                       /* Loop on values for this attribute */
                       Enumeration enumVals = anAttr.getStringValues();
                       while ( enumVals.hasMoreElements() ) {
                           String aVal = ( String )enumVals.nextElement();
                           System.out.println( "" + aVal );
                       }
                   }
               }
           }
           catch( LDAPException e ) {
               System.out.println( "Error: " + e.toString() );
           }
           /* Done, so disconnect */
           if ( ld.isConnected() )
               ld.disconnect();
       }
   }
```

```
   import netscape.ldap.*;
   import java.util.*;

   public class ModifyEmail {
       public static void main( String[] args )
       {
           try {
               LDAPConnection ld = new LDAPConnection();
               /* Connect to server */
               String MY_HOST = "localhost";
               int MY_PORT = 389;
               ld.connect( MY_HOST, MY_PORT );

               /* authenticate to the directory as Bab Jensen */
               String MY_NAME = "cn=Barbara Jensen,o=Ace Industry,c=US";
               String MY_PASSWORD = "MysteryLady";
               ld.authenticate( MY_NAME, MY_PASSWORD );

               /* Prepare to change my email address */
               LDAPAttribute attrEmail =
                       new LDAPAttribute( "mail", "babs@ace.com" );
               LDAPModification mod =
                       new LDAPModification( LDAPModification.REPLACE,
                                             attrEmail );

               /* Now modify the entry in the directory */
               ld.modify( MY_NAME, mod );
               System.out.println( "Entry modified"  );

           }
           catch( LDAPException e ) {
               System.out.println( "Error: " + e.toString() );
           }
           /* Done, so disconnect */
           if ( ld.isConnected() )
               ld.disconnect();
       }
   }
```

11. Appendix B - Changes from draft-ietf-asid-ldap-java-api-00.txt

11.1. LDAPConnection

The method setProperty() throws an LDAPException if the specified pro-
perty is not supported.

11.2. Controls

A section was added with support for currently defined LDAP controls.

11.3. LDAPException

Constructors LDAPException(String) and LDAPException(String,int,String)
were removed, and the constructor LDAPException(String,int) was added.
The description stating that the resultCode is no longer valid was
removed.

11.4. LDAPSearchResults

The method next() throws an LDAPReferralException if automatic referral
following is not enabled.

The getBaseAttribute() method was removed.

The getAttribute(attrName) method was redefined to return only an exact
match, and a new method getAttribute(attrName, lang) was added to pro-
vide language-sensitive best-match functionality. Neither method throws
an exception.

11.5. LDAPSecurityException

The class was removed.

11.6. LDAPSortKey

The class was added, to support server-side sorting.

11.7. LDAPv3

The PREFERRED_LANGUAGE option was added to setOption().

11.8. Examples

The disconnect() was moved to after the exception catcher. An example
of using paged sort controls was added.

12. Appendix C - Changes from draft-weltman-java-ldap-00.txt

12.1. LDAPv3

This interface is new. It adds support for features of LDAP protocol version 3.

12.2. SSL -> TLS

References to the Secure Socket Layer (SSL) have been replaced with references to Transport Layer Security (TLS).

12.3. LDAPAttributeSet

Methods getAttribute(String name) and getBaseAttribute(String name) were added.

12.4. LDAPCompareAttrNames

Constructors were added to allow specifying sorting in descending, and not just ascending, order.

12.5. LDAPCompareAttrNames

Constructors were added to allow specifying sorting in descending, and not just ascending, order.

12.6. LDAPControl

This is a new class to support the LDAPv3 protocol extension, where Server or Client Controls may be specified for LDAP operations.

12.7. LDAPEntry

Methods getAttribute(String name) and getBaseAttribute(String name) were added.

12.8. LDAPException

The method getMatchedDN was added.

12.9. LDAPExtendedOperation

New class to pass extended operations back and forth to/from the server, for LDAPv3.

12.10. LDAPv2

For connect(), the "host" parameter may be a space-delimited list of hosts to attempt to connect to. Each one may have a colon and a port number attached.

12.11. Dereferencing aliases

LDAPConnection.setOption(), the LDAPSearchConstraints constructor, LDAPSearchConstraints.getDereference(), and LDAPSearchConstraints.setDereference() were changed so that the option specifying how to dereference aliases is now an integer instead of a boolean, and the legal values are declared.

12.12. Default referral hop limit

The default referral hop limit in LDAPConnection.setOption() was changed from 5 to 10.

12.13. Examples

An example of how to modify an existing Directory entry was added to appendix A.

13. Appendix D - Outstanding issues

13.1. Support for SASL authentication

The framework suggested in the LDAPv3 interface for SASL authentication is tentative. It will need to be extended to and integrated into the automatic referral-processing architecture, so that an LDAPRebindProc can initiate new authentication procedures with servers that are referred to by a search. The current specification only allows for simple authentication on automatic referral following.

If referrals are handled explicitly rather than automatically, by catching LDAPReferralException, the caller may use the LDAPv3 SASL framework.